E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Detección automática de pólipos basada en el Histograma de Gradientes Orientados

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Iván Del Burgo Ullate

Director y codirector: Daniel Paternáin y Aránzazu Jurío

Pamplona, 31-05-2017

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen:

La detección automática de pólipos consiste en localizar de la manera más exacta posible la localización de un pólipo en una imagen colorrectal. De esta forma, se pretende servir de ayuda a los profesionales médicos que realizan el diagnóstico de manera visual.

En este proyecto se pretende investigar la detección de pólipos mediante la combinación de técnicas de visión artificial y técnicas inteligentes.

Se pretende estudiar la técnica de extracción de características sobre imágenes colorrectales, en este caso, el histograma de gradientes orientados. Estas técnicas nos permiten extraer características de los pixeles que nos ayudan en la tarea de clasificar un pixel como perteneciente al pólipo o no. Para la clasificación podríamos utilizar técnicas relacionados con la Minería de Datos, utilizando algún clasificador como pueden ser la SVM (Máquinas de soporte Vectorial) o las redes neuronales.

Palabras Clave: Visión artificial, pólipos, clasificación, redes neuronales, histograma de gradientes orientados.

1 Introducción y objetivos

En este trabajo se plantea un problema que está cada vez más al alza. Atacando este problema se pretende ayudar a los profesionales de la medicina en su labor mediante técnicas computacionales e inteligencia artificial, ya sea para prever cualquier tipo de patología mediante el uso de minería de datos o para ayudar al médico alertándole de algún posible diagnóstico. El uso de éstas técnicas en el ámbito médico es una manera de dotar de cierta objetividad al posible diagnóstico, ya que un profesional médico, al ser humano puede tener días mejores y días peores, ya sea por cansancio o cualquier otro tipo de problema, y esto, condicionaría su diagnóstico en cierta manera. Sin embargo, un computador no se cansa ni tiene problemas, y siempre dará los mismos resultados para las mismas entradas, por lo que se considera más objetivo, aunque no deje de ser eso, una ayuda al profesional médico, y será él el que tenga la decisión final.

En concreto, el problema al que se va a intentar dar solución va a ser la búsqueda de pólipos en imágenes colorrectales mediante técnicas de visión artificial. Es decir, dada una imagen, se intentará encontrar el posible pólipo, si lo hay, contenido en ella para ayudar al médico en su labor. Este tema es muy interesante, ya que se trata de un problema médico que permitirá ayudar a personas reales si se implanta a nivel profesional. El objetivo final es conseguir un método lo suficientemente rápido como para poder obtener la ubicación del pólipo en una imagen casi en tiempo real, ya que la obtención de estas imágenes se hace mediante vídeo y se conseguiría analizar cada frame del video para ver el pólipo mientras se realiza el examen visual.

Para solucionar este problema planteado vamos a utilizar un conjunto de técnicas que consisten en la extracción de características de una imagen y su posterior clasificación que nos permita identificar el hecho que estamos buscando. Estas técnicas se han utilizado para la localización de diversos elementos en imágenes, aunque principalmente se han utilizado para la localización de peatones.

Analizando el problema al que se va a intentar dar solución, se pueden discernir dos grandes áreas de conocimiento. La primera área es el área de la visión artificial, ya que como se ha dicho anteriormente, lo que se quiere es extraer información a partir de una imagen de entrada. La otra gran área es la minería de datos, pues al extraer las características de la imagen se va a disponer de una gran cantidad de datos, y para analizarlos y conseguir la información necesaria se van a emplear técnicas relacionadas con éste área.

El procesamiento digital de imágenes o visión artificial, es un conjunto de técnicas que permiten mejorar la calidad o facilitar la búsqueda de información en una imagen. Estas técnicas están siendo utilizadas en la actualidad en multitud de ámbitos. Desde el entorno empresarial para un reconocimiento facial, pasando por el entorno médico en la ayuda de los profesionales del ámbito.

Estas técnicas permiten reconocer patrones, formas y otros elementos que se asemejarían a lo que una persona podría reconocer en una imagen. Por ejemplo, separar un objeto del fondo, aunque si bien es cierto que se pueden realizar este tipo de técnicas sin tener una visión concreta del problema, muchas veces, el conocer a fondo el problema puede ayudar a discernir un punto esencial para que la técnica a emplear funcione mejor.

Véase por ejemplo un problema de segmentación en el que queremos contar objetos encima de una superficie se podría encontrar el fondo de la superficie y establecer eso como fondo y todo

lo demás como objetos. Si dentro de ese problema, sabemos que lo que se va a contar son monedas encima de la superficie, podríamos quitar muchos de esos objetos reconocidos tan solo exigiendo que los objetos encontrados tengan una forma redondeada.

Por otro lado, la minería de datos. Es una de las etapas del Knowledge Discovery in Databases (KDD), el KDD es un campo de la estadística y las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de conjuntos de datos. Utiliza métodos de la inteligencia artificial, de la estadística y del aprendizaje automático.

El objetivo de la minería de datos consiste en extraer la información útil de una gran cantidad de datos y transformarla de tal manera que sea más comprensible o sencilla para su uso posterior.

2 Pasos previos

2.1 Investigación médica

Para poder aproximar una solución al problema lo más certera posible, el primer paso que se ha tenido que realizar ha sido una profunda investigación de las características de los pólipos, ya que existen múltiples maneras de abarcar el problema.

Tras revisar diversos articulos e información médica sobre los pólipos se vio que había al menos dos distintas formas de identificar un pólipo.

1. **Basada en su histología (información microscópica):** Existen 5 tipos de pólipos.
    a. Adenoma (epitelial)
    b. Pólipo neoplásico no epitelial.
    c. Pólipo hamartomatoso
    d. Pólipo inflamatorio
    e. Pólipo hiperplásico
2. **Basada en la forma (información macroscópica):** Existen 2 tipos.
    a. Pólipo sésil
    b. Pólipo pediculado

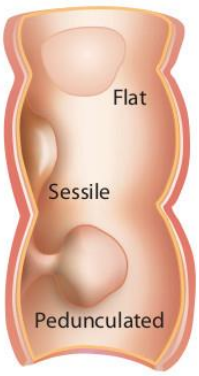


Fig. 1. Imagen que muestra la diferencia entre los tipos de pólipos. Sésil (Sessile), no tiene tallo sino que está pegado a la pared. Y pediculado (Pedunculated), tiene un tallo que lo despega de la pared y una protuberancia al final del tallo.

Esta parte es una de las partes más importantes de un proyecto, ya que para poder realizar el proyecto con precisión es importante saber y dominar el tema que se está tratando. De esta manera surgen nuevas aproximaciones al problema y nuevos métodos que pueden ayudar a resolverlo.

Un pólipo es una protuberancia circunscrita visible macroscópicamente que se proyecta en la superficie de una mucosa, en este caso el colon.

El cáncer de colon es el tumor maligno de mayor incidencia en España. El riesgo de padecer cáncer de colon es de un 5% en los hombres y de un 3.3% en las mujeres en España según la asociación española contra el cáncer (AECC). Tiene una tasa de supervivencia de un 64% a los 5 años. Éste es originado por un pólipo que en su inicio podía ser benigno y que posteriormente evolucionó hasta convertirse en canceroso.

Este tipo de cáncer es fácilmente previsible, y su previsión consiste en un seguimiento mediante distintas pruebas, entre ellas la colonoscopia, a partir de los 50 años o en menor edad si el paciente tiene mayor riesgo de contraer este tipo de cáncer. Se estima que desde el momento en que las primeras células comienzan a alterarse para dar lugar a un pólipo, hasta que éste se convierte en canceroso pueden transcurrir entre 10 y 15 años. Esto quiere decir que con un correcto sistema de prevención, se puede evitar el cáncer de colon en la mayoría de ocasiones.

El tratamiento a aplicar una vez se encuentra un pólipo consiste en su extirpación mediante polipectomía endoscópica en caso de que la situación lo permita (situación del pólipo, tamaño, malignidad,…). Esta técnica consiste en extirpar el pólipo del cuerpo y su posterior cauterización. Es una operación sencilla y no suele causar complicaciones. En otros casos, se podría extirpar mediante cirugía.

2.2 Búsqueda de la base de datos

Para poder realizar este proyecto ha sido necesaria una base de datos de imágenes colorrectales. Estas bases de datos son muy complicadas de conseguir, ya que se están tratando imágenes médicas de pacientes reales, por lo que no son bases de datos que se puedan encontrar fácilmente, y las que se encuentran no son muy extensas.

Durante las lecturas de diversos artículos, los cuales trataban sobre temas relacionados con la medicina también, se mencionan en numerosas ocasiones las bases de datos que utilizan para poder efectuar su solución al problema que les plantean. Estas bases de datos que mencionan no estaban disponibles para su utilización.

Debido a esto se tuvo que encontrar una base de datos que estuviera accesible para su utilización. Finalmente se encontró una base de datos de un grupo de investigación sobre el tema. Esta base de datos ha sido obtenida de un grupo de investigación de Barcelona denominado Machine Vision Group, que se centra en diversos temas relacionados con la visión artificial. Este grupo intenta conseguir un algoritmo de búsqueda de pólipos en colonoscopias en tiempo real. Su base de datos es accesible mediante su previa petición.

Esta base de datos contiene la información de 15 secuencias de vídeo, separadas en imágenes. Cada una de estas imágenes representa un frame del video. En concreto contiene 1140 imágenes, divididas en 3 secciones de igual tamaño, por lo que cada sección contiene 380 imágenes. De las 3 secciones, la primera contiene los frames del vídeo sin procesar, mientras que las otras dos secciones contienen imágenes procesadas por un médico de manera manual. Uno de los objetivos de este proyecto es ser capaces de realizar ese proceso de manera automática. Sin embargo, puesto que nuestro algoritmo necesita aprender a identificar los pólipos, se necesita un conjunto como es éste que contenga un grupo de imágenes ya segmentado. Por cada frame del vídeo hay una imagen en cada sección, con distinta información en cada una de ellas:

1. **Sección de imágenes reales**: Esta sección contiene la imagen real de la colonoscopia, es decir cada uno de los frames.

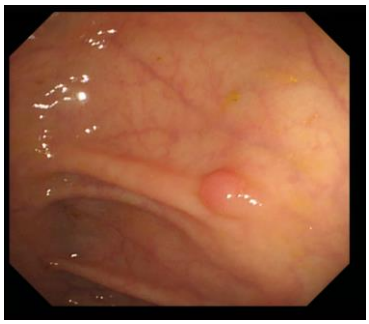


Fig. 2. Imagen de la base de datos del primer tipo. Muestra un frame de una de las secuencias de vídeo de los que se dispone.

2. **Sección de bordes de los pólipos**: En esta sección se disponen de imágenes en las que se aprecia el borde de cada pólipo en cuestión.

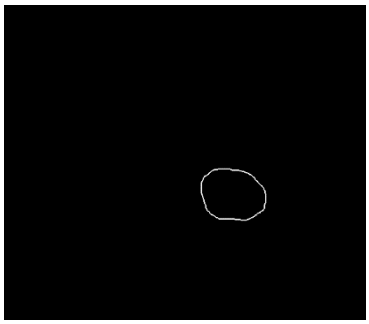


Fig. 3. Imagen de la base de datos del segundo tipo. Muestra un frame de una de las secuencias de vídeo de los que se dispone procesado para obtener el borde del pólipo presente en la frame del primero de los tipos asociado.

3. **Sección de segmentación de los pólipos**: En esta sección se encuentran imágenes segmentadas que contienen la información del pólipo completa, no sólo los bordes.

Fig. 4. Imagen de la base de datos del tercer tipo. Muestra un frame de una de las secuencias de vídeo de los que se dispone procesado para obtener el pólipo relleno presente en la frame del primero de los tipos asociado.

2.3 Definición del problema

Una vez leídos los papers y visto que la mejor forma de detectar todos los posibles pólipos que haya presentes en la imagen es, en este caso, según su forma, hubo que establecer el cómo se iba a realizar el proyecto.

Para este proyecto, se van a dividir las imágenes en otras más pequeñas (en nuestro proyecto utilizaremos 64x64 píxeles) y clasificar estas imágenes más pequeñas como imágenes independientes para ver si contienen un pólipo o no. En este problema queremos centrarnos principalmente en las formas que tienen los pólipos. Estas formas son características de los pólipos, ya que son formas ovaladas sobre imágenes con pocos objetos de esta forma.

Por ello, se han estudiado otros trabajos sobre problemas de encontrar objetos basándose en las formas de las imágenes. Algunos de estos trabajos podían ser proyectos para encontrar caras [1] y encontrar peatones [2] en imágenes mayoritariamente. Se estableció que en este caso, se quería encontrar la forma de los pólipos, en este problema tienen una forma más o menos ovalada.

Para buscar la forma de los pólipos en cada subimagen o ventana, se transforma la información contenida en las intensidades de los píxeles en otro tipo de características obtenidas a partir de la información de las intensidades, que nos proporciona la información de las formas. En este caso se va a usar el extractor de características denominado como HOG (Histogram of oriented gradientes) o histograma de gradientes orientados. Este extractor consiste en crear un vector de características que contenga la información sobre las formas presentes en la imagen.

Para encontrar los pólipos, se extraen las características de las subimágenes que definen las formas que hay en ella y posteriormente se introducen estas características en un clasificador entrenado, y será éste el que confirme si se trata de una imagen que contenga un pólipo o no.

3 Entorno de trabajo. Matlab

En un principio se plantearon dos posibles entornos, Python y MatLab. Python es un lenguaje de programación que cada vez está siendo más utilizado en el entorno de la computación y la inteligencia artificial, debido a su potencia de cálculo y la cantidad de módulos de los que se dispone que facilitan bastante el trabajo.

Por otro lado tenemos el entorno de MatLab (Matrix Laboratory), este entorno es uno de los que más se utiliza para el desarrollo de temas relacionados con la inteligencia artificial y la computación ya que muy comúnmente las matemáticas computacionales utilizan las matrices como unidad de trabajo, y éste entorno está especialmente diseñado para trabajar con ellas por lo que su usabilidad con este tipo de estructura de datos es muy alta. Todo está orientado a trabajar con este tipo de estructuras y las imágenes no son otra cosa que esto, matrices de 3 dimensiones, en las que se dispone de ancho, alto y en nuestro caso la tercera dimensión indica los tres colores de la codificación de color RGB (Rojo, verde y azul respectivamente).

Todo esto hacía que me resultase más sencillo trabajar con MatLab. Además a lo largo de todo el grado ha sido el lenguaje que he utilizado en las distintas asignaturas de la mención de computación, por lo que no era necesario aprender este lenguaje debido a que ya lo conocía bastante. Por otro lado, MatLab dispone de muchas funcionalidades ya creadas y muy optimizadas para su utilización. Debido a esto ha sido el lenguaje elegido para este proyecto.

Aunque finalmente se eligió MatLab como entorno de realización, en un futuro pretendo migrar todo el código que se ha creado en Matlab al lenguaje de Python y comprobar el rendimiento que éste lenguaje ofrece.

4 Pseudo-algoritmo

1. Creación de la base de datos
    a. Convertir el conjunto de imágenes completo a escala de grises
    b. Preprocesar cada una de las imágenes para corregir los píxeles negros y eliminar los brillos presentes
    c. Extraer las ventanas (subimágenes) de la imagen que contienen pólipos, para tratarlas como muestra positivas
    d. Extraer algunas ventanas de la imagen que no contienen pólipos para tratarlas como muestras negativas
2. Entrenamiento del clasificador
    a. Extraer las características de cada imagen de entrenamiento.
    b. Entrenar una SVM (Supporting Vector Machine) con el conjunto de características extraído de cada imagen.
    c. Entrenar una o varias redes neuronales con el conjunto de características extraído de cada imagen.
3. Clasificación de imagen
    a. Recorrer cada imagen con una ventana deslizante de 64x64 píxeles y clasificarla
        i. Extraer las características de la ventana.
        ii. Introducir las características obtenidas como entrada a los clasificadores.
        iii. Introducir la respuesta de los clasificadores en la matriz de resultado.
4. Post-proceso
    a. Limpiar la imagen resultado de aquellos puntos aislados.
    b. Dibujar recuadros alrededor de los puntos para presentar de una manera más visible los resultados.

5 Procedimiento

5.1 Creación de Base de Datos

La base de datos de la que se dispone consiste en 380 imágenes colorrectales que corresponden a 15 secuencias de vídeo. Además de estas imágenes, se tienen 380 imágenes binarizadas que corresponden a los contornos de los pólipos presentes en la imagen a la que corresponden, y además, otras 380 imágenes también binarizadas que indican toda la forma del pólipo.

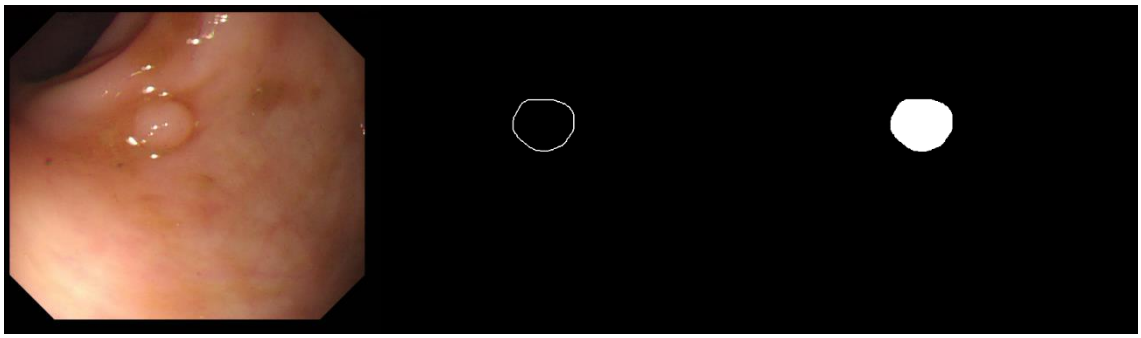


Fig. 5. Imagen que muestra los tres tipos de imágenes de los que se dispone. A la izquierda se ve el primer tipo, que corresponde a la imagen sacada del vídeo sin ningún procesado. En el medio se ve el segundo tipo de imágenes, que contiene la segmentación del borde del pólipo que se ve en su asociada al primer tipo. Y el tercer tipo a la derecha que contiene la información del pólipo completo presente en la imagen asociada del primer tipo.

En la figura 5 se puede ver claramente los tres tipos de imágenes de las que se disponen. A la izquierda, la imagen de la secuencia de vídeo sin procesar que se ha mencionado anteriormente. En el centro, la imagen binarizada que muestra el contorno y por último, a la derecha, la imagen que muestra el pólipo relleno.

En este proyecto se clasifican las imágenes mediante el procedimiento de clasificación de todas las posibles ventanas, es decir, dividir la imagen en imágenes más pequeñas (subimágenes) que en nuestro caso serán de 64x64 píxeles y clasificarlas de forma independiente. Por tanto, para hacer la clasificación, que será supervisada, necesitamos un conjunto de entrenamiento que contenga tanto muestras positivas como muestras negativas. Cada uno de los ejemplos del conjunto de entrenamiento se extrae de las imágenes de la base de datos obtenida, es decir de los frames de las secuencias de vídeo que tenemos.

Con la información de los otros dos tipos de frames se puede saber qué tipo de ejemplo estamos extrayendo, si se trata de un ejemplo de entrenamiento positivo o ejemplo de entrenamiento negativo. Cuando se han extraído todos los ejemplos, se le proporcionan al clasificador para su entrenamiento.

En total se extraen un pequeño conjunto de imágenes de entrenamiento que sí contienen pólipo en ellas, que llamaremos conjunto de entrenamiento positivo, y otro conjunto bastante más amplio de imágenes que no contienen pólipo en ellas, que llamaremos conjunto de entrenamiento negativo. Ambos casos se almacenan en directorios diferentes apropiadamente nombrados para poder identificar cada uno de estos conjuntos. Todas las imágenes procesadas se han almacenado en un formato de color de escala de grises.

Para crear la base de datos primero se decidió que el conjunto de entrenamiento positivo tenía que contener la imagen de 64x64 que contiene el pólipo entero y las 8 imágenes de alrededor suyo.

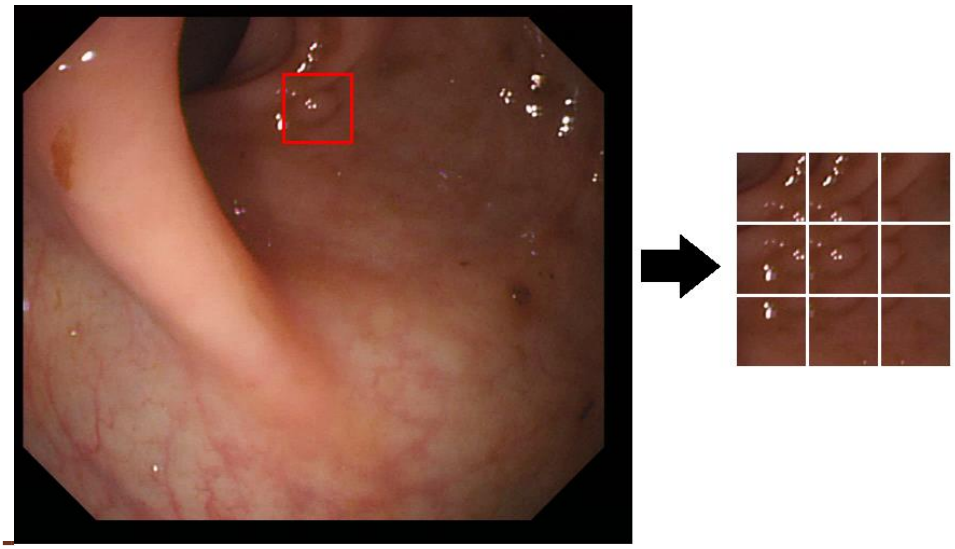


Fig. 6. Ejemplo de extracción de las 9 ventanas que corresponden a muestras positivas. En rojo se puede ver el pólipo recuadrado (64x64 píxeles) y de esa ventana, deslizándola 32 píxeles en todas las direcciones salen las otras 8 ventanas que se pueden ver a la derecha.

Tal y como se muestra en la figura 6, el pólipo está recuadrado en rojo, y a partir de esta ventana de 64x64 píxeles, se obtienen las otras 8 ventanas de 64x64 pixeles también, deslizando cada vez 32 píxeles en las cuatro direcciones.

Posteriormente se comprobó que esto provocaba bastantes falsos positivos, ya que al entrenar el clasificador se entrenaba con imágenes positivas que contenían un pequeño pliegue nada más y llevaba a error porque la imagen contiene pliegues en el colon que podrían parecer al clasificarlas el borde de un pólipo, haciendo que el entrenamiento sea infructuoso.

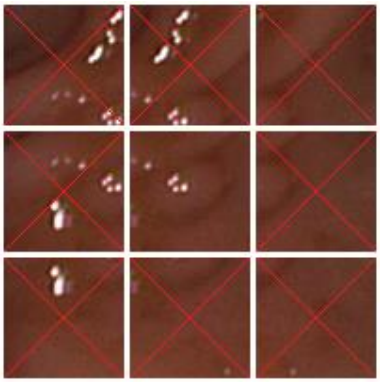


Fig. 7. En esta figura se ve que de las 9 subimágenes que se extraen solo nos quedamos finalmente con la del medio, que es la que corresponde al pólipo completo.

Debido a este problema y para solucionar casos de falsos positivos, se suprimieron las 8 imágenes de alrededor del pólipo que eran aquellas que hacían que el entrenamiento produjera estos errores. De este modo tan solo quedaban aquellas imágenes que contenían el pólipo casi entero o entero en la ventana de 64x64 píxeles antes mencionada.

Finalmente, se concluyó que lo mejor era que el pólipo estuviese contenido completamente en la ventana de 64x64. Ya que si no, la ventana que se dispone (64x64) no contendría información sobre la forma suficiente, debido a que los bordes del pólipo quedarían fuera de ésta. Para ello se utilizó la información de la que se dispone en las otras imágenes, en concreto las que se han mencionado anteriormente, que habían sido procesadas de manera manual por un médico, para crear una tabla de verdad que nos indica dónde están los pólipos en las imágenes y además qué imágenes contienen pólipos cuyo tamaño se ajuste a una ventana de 64x64.

De esta manera se ha podido automatizar el proceso de extracción de muestras positivas para el clasificador. Las muestras negativas se han extraído de forma manual.

Tras todo el proceso de la extracción de las muestras positivas y negativas, se consigue construir un dataset que contiene.

- 379 imágenes de test de las que se extraen las muestras positivas y negativas. De 500x574 píxeles.
- 26 imágenes de 64x64 píxeles con pólipos (muestras positivas)
- 480 imágenes de 64x64 píxeles sin pólipos (muestras negativas)

5.2 Pre procesamiento de imagen

En la primera aproximación al problema no se hacía ningún tipo de tratamiento a las imágenes, por lo que las imágenes que se utilizaban contenían diversos elementos que hacían más difícil su clasificación. El elemento que más dificultaba la clasificación eran los brillos presentes en la imagen. Estos brillos están presentes debido a que las imágenes están hechas en el interior de un cuerpo humano, por lo que para ver algo hay que iluminar allí por donde se pasa y el colon es una parte del cuerpo que contiene mucho líquido y esto hace que la luz emitida por la cámara para poder visualizar el interior refleje en el líquido y se produzcan estos brillos.

Los brillos además tienen una forma ovalada, bastante parecida a un pólipo. Esto hacia que muchos de los brillos fueran clasificados erróneamente por el algoritmo.

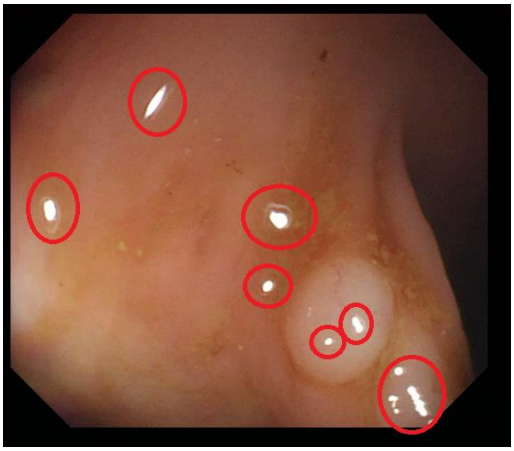


Fig. 8. Imagen que muestra la ubicación de varios brillos presentes en ella mediante su redondeado con círculos rojos.

5.2.1 Extracción de brillos

El primer paso que hubo que dar para mejorar el algoritmo y evitar los problemas ocasionados por los brillos fue tratar de eliminarlos. Esta tarea, a pesar de parecer una tarea sencilla ha sido una de las que más tiempo me ha ocupado, ya que debido a la naturaleza de las imágenes ha sido muy complicada de terminar eficazmente.

Para poder llevar a cabo todos los algoritmos de eliminación de los brillos que se van a explicar a continuación, previamente hay que encontrar dónde están los brillos. Para ello se empleó un umbral (de 0.82 en una escala de 0 a 1), y si un pixel sobrepasa ese umbral se considerará que es lo suficientemente blanco como para considerarlo brillo. Este umbral fue elegido acorde con las imágenes ya que no solo había que encontrar el brillo en sí, sino también una pequeña zona alrededor del mismo.

Posteriormente, a todos aquellos píxeles donde se considere que hay un brillo se les aplica una dilatación para que se amplíe la zona donde se ha encontrado como brillo. La razón de esta dilatación es que en las imágenes que tenemos, los brillos en numerosas ocasiones se encuentran rodeados por una "aureola" de luz, es decir que alrededor de un brillo existe un pequeño trozo de la imagen que se ha aclarado debido al brillo.

Con esta información se construyen dos imágenes. La imagen en escala de grises real, y otra imagen binarizada que contiene la información necesaria para saber dónde se encuentran los brillos presentes en la imagen.

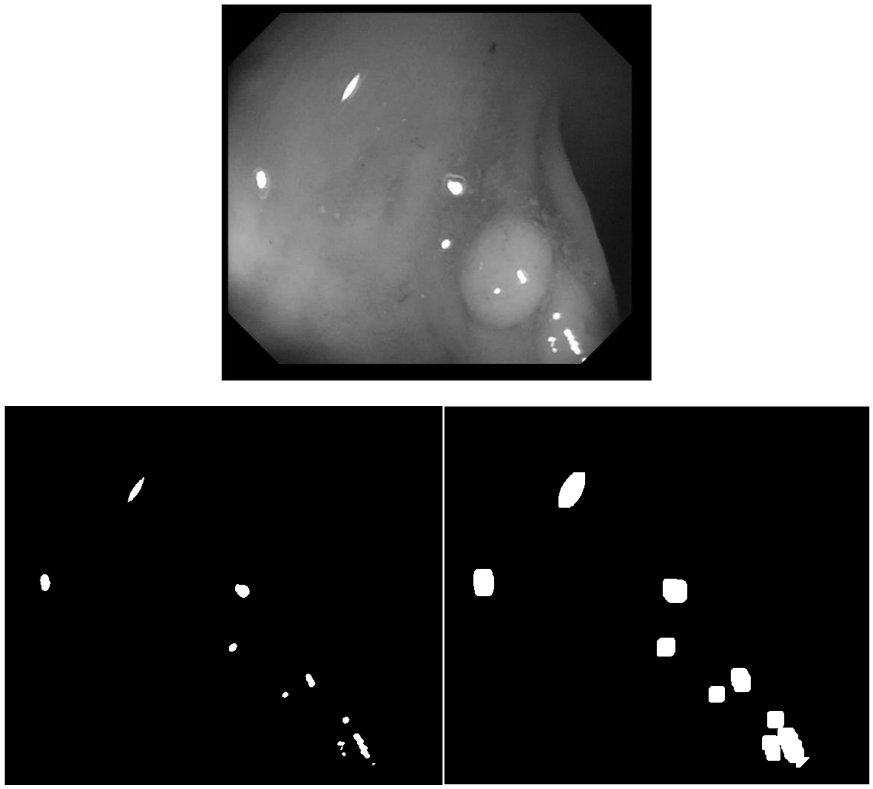


Fig. 9. En la figura se puede ver la imagen original (arriba) junto con su proceso de búsqueda de brillos, la imagen segmentada de brillos sin dilatar (abajo a la izquierda) y la misma imagen con los brillos dilatados (abajo a la derecha).

5.2.1.1 Inpainting

Tras el primer paso de segmentar la imagen, para obtener otra, que contenga la información de la posición y tamaño de los brillos presentes, lo que se pretende es encontrar la manera de rellenar aquellas zonas que la imagen segmentada marque como brillo. Para ello se emplea la técnica de inpainting.

El inpainting, es un proceso que permite recuperar una parte deteriorada de una imagen o alguna zona que contenga un objeto a eliminar con el objetivo de mejorar su calidad. Es decir, lo que se pretende es recuperar una información desaparecida de la imagen a través de algoritmos que calculen esa información a introducir mediante la utilización de la información del entorno. En este proyecto se han utilizado varias de estas técnicas para el cálculo de la información perdida. En concreto, se han utilizado 3 técnicas que serán explicadas a continuación:

- Algoritmo de la media
- Algoritmo de Criminisi
- Algoritmo de la cebolla

5.2.1.1.1 Algoritmo de la media

La primera aproximación que llevé a cabo es un algoritmo muy sencillo, que denominaremos como 'Algoritmo de la media'. De lo que trata es de procesar la imagen para que se rellene el espacio que se ha encontrado como brillo mediante un proceso sencillo y que no consuma mucho tiempo de ejecución, ya que no era el motivo principal del trabajo.

Este algoritmo consiste en que a partir de la imagen segmentada en la que tenemos la información de los brillos, coger todos los píxeles que hemos segmentado como brillo y sustituirlos por otros, que se calculan a través de la información de los píxeles de alrededor.

La forma de sustituir el píxel del brillo que se está teniendo en cuenta en cada iteración del algoritmo es calcular la media de los píxeles de alrededor, que no correspondan al brillo. Para ello siempre se empieza a sustituir por el primer píxel de la imagen binarizada que MatLab encuentre en blanco, por lo que siempre empieza de arriba abajo y de izquierda a derecha. Esto hace que la información que hay a la izquierda del brillo sea muy relevante, ya que al utilizarlo hace que la información de las intensidades se extienda desde la izquierda.

5.2.1.1.2 Algoritmo de cebolla

Este es el segundo algoritmo que se utiliza en el proyecto. En este algoritmo se tiene en cuenta qué pixel de los que hay que rellenar tiene más información a su alrededor, el método que se utiliza para calcular la información que se va a introducir es hacer la media de los pixeles conocidos, al igual que en el algoritmo de la media. Es decir, se va rellenando aquel píxel que tenga más píxeles conocidos a su alrededor. De esta manera, se evita el problema del orden de rellenado del algoritmo de la media, y se rellenan de una manera más satisfactoria los brillos.

5.2.1.1.3 Algoritmo de Criminisi

El tercer algoritmo que preparé para este proyecto es el de inpainting propuesto por Criminisi [3]. Es un excelente algoritmo de relleno en caso de querer mantener las texturas al rellenar. Este algoritmo es el de rellenado de brillos más complicado que se ha implementado en este proyecto, y se diferencia de los otros dos en que trabaja por bloques.

Este algoritmo utiliza el gradiente para rellenar primero aquellas zonas donde el cambio de intensidad es mayor, por ejemplo, en caso de tener una línea en el medio de la imagen, hay que intentar primero rellenar la línea y posteriormente rellenar el resto de la imagen que haya para así mantener la forma de la imagen original.

Para ello es necesario calcular unas prioridades a cada pixel, y empezar a rellenar por el píxel de mayor prioridad.

Para calcular las prioridades, se calcula el gradiente junto con la dirección de éste, tanto de la imagen binarizada como de la original. Para cada pixel de la imagen tenemos una confianza, que comenzará en 1 para los píxeles que no queremos modificar de la imagen segmentada y 0 para los píxeles que corresponden a una zona de brillo. A esta matriz de confianzas le llamaremos C.

Por otro lado, dispondremos de una matriz que llamaremos D, que será la que compute la información que hemos mencionado anteriormente que permita comenzar por aquellos píxeles que necesiten ir primero (líneas, formas, etc.). La multiplicación de la C y la D nos proporciona la matriz de prioridades para cada píxel.

Seleccionamos el píxel de mayor prioridad y cogemos una ventana de 9x9 con ese píxel en el centro. Lo que se hace es buscar la ventana más parecida (sin tener en cuenta el brillo) a la ventana que se está evaluando. Una vez se haya encontrado la ventana más parecida se sustituye la información que falte (el brillo) por aquella de la ventana encontrada. Se recalculan tanto la C como la D, pero esta vez de la imagen que se acaba de calcular, es decir de la imagen en la que se acaba de introducir información que no se tenía en la iteración anterior y se vuelven a calcular las prioridades, haciendo así que se repita el proceso hasta haber rellenado el total de los huecos.

Si tenemos la siguiente imagen que corresponde al píxel que se quiere rellenar (en rojo, con el algoritmo de inpainting propuesto por Criminisi. Asumiendo que la zona blanca es parte del brillo encontrado. Compararíamos esta ventana de 9x9 píxeles con todas las posibles ventanas de la imagen.

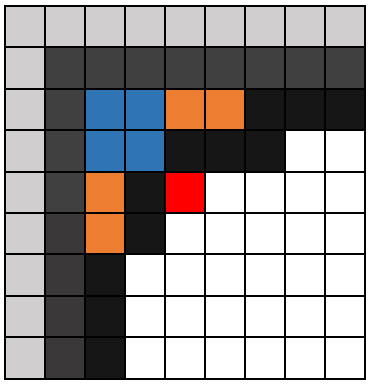


Fig. 10. Ventana de 9x9 correspondiente al algoritmo de Criminisi que contiene una parte del brillo que se desea rellenar.

Sean por ejemplo estas otras dos ventanas las posibles a elegir:

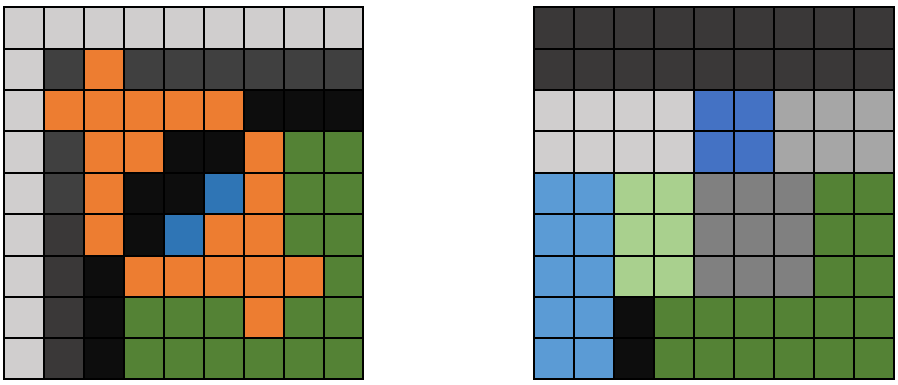


Fig. 11. Dos posibles ventanas que pueden ser elegidas como las más parecidas. Se observa que la ventana de la izquierda es más parecida a la figura anterior que la ventana de la derecha.

Se calcula cuál de ellas es la ventana más parecida a la que buscamos. Y se establece que la más parecida es la ventana que tenemos a la izquierda. Por lo que es la ventana que se va a introducir en la imagen. La parte que se va a rellenar es tan solo aquella que pertenece al brillo, es decir, la parte que en la primera imagen tenemos en blanco.

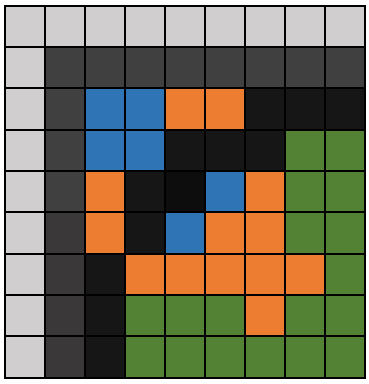


Fig. 12. Esta ventana muestra cómo quedaría la ventana final tras introducir la parte que se desconocía perteneciente al brillo. Se puede observar que no se cambia la ventana entera, sino solo aquella parte que pertenecía al brillo.

5.2.1.2 Ejemplo

A continuación se detalla un ejemplo de los tres métodos de inpainting utilizados para clarificar su utilización y proceso.

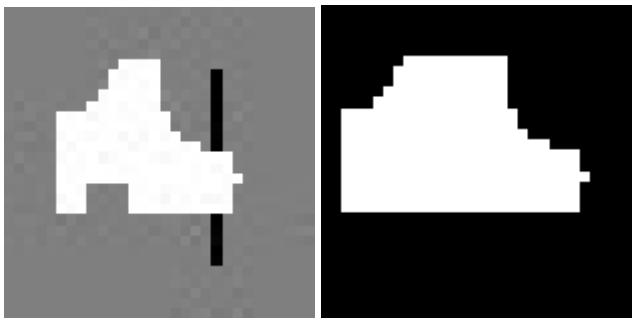


Fig. 13. En la figura se pude observar un ejemplo de imagen que contiene una línea negra (izquierda) que ha sido cortada mediante la introducida de forma manual para seleccionar la forma a rellenar. También se puede observar la imagen segmentada (derecha) con lo que el algoritmo interpreta como brillo y su proceso de dilatado.

Con estas dos imágenes superiores de la figura 13, que serían iguales para los tres algoritmos, se procede a intentar su rellenado, cada uno de los algoritmos empezará a rellenar según su criterio. A continuación muestro un ejemplo de cómo funcionan los tres métodos de inpainting que he implementado sobre una misma imagen. En la imagen superior se observa la imagen original (a la izquierda) de la que se parte para rellenar la zona blanca, a su lado, a la derecha, se observa la misma imagen donde se ha segmentado y dilatado el brillo. Todos los píxeles blancos de esta imagen segmentada son los que se van a rellenar mediante las tres técnicas explicadas.

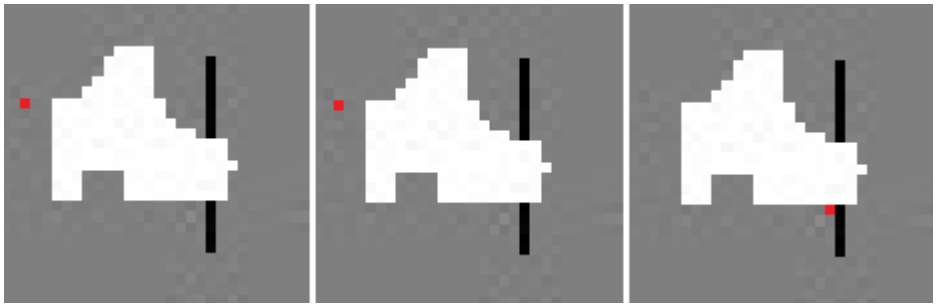


Fig. 14. En la figura se puede observar el píxel por el que comienza a rellenar cada uno de los algoritmos en rojo. El algoritmo de la media a la izquierda, el de cebolla en el centro y el propuesto por Criminisi a la derecha.

El pixel rojo indica la posición desde la que empieza a rellenar cada uno de los algoritmos, el algoritmo de la media, el algoritmo de cebolla y el algoritmo propuesto por Criminisi respectivamente. Se observa que el algoritmo propuesto por Criminisi comienza a rellenar alrededor de la línea negra, ya que intenta preservar estos elementos empezando por ellos. También se puede observar que el algoritmo de la media y el de la cebolla han comenzado por el mismo pixel sin embargo en su funcionamiento posterior se puede comprobar que el algoritmo de la media recorrerá la imagen de arriba hacia abajo y de izquierda hacia la derecha mientras que el de la cebolla irá rellenando por capas hacia el centro.

Tras unas pocas iteraciones se puede ver el orden en el que se van rellenando cada una de las imágenes dependiendo del algoritmo.

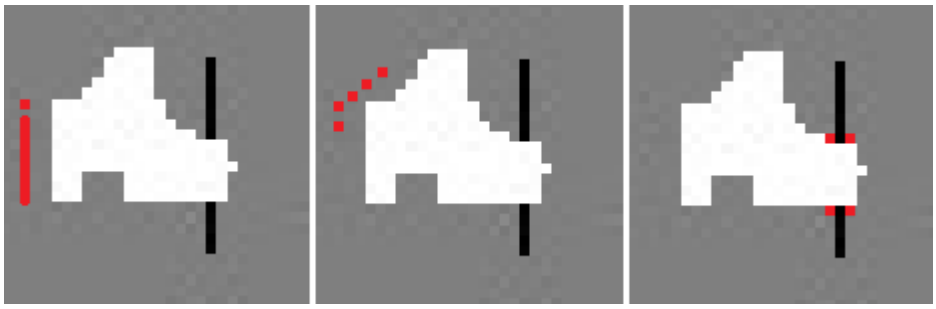


Fig. 15. Figura que muestra el orden de rellenado de cada uno de los algoritmos. El de la media (izquierda) comienza de arriba abajo por la izquierda. El de cebolla (medio) va por capas. Y el propuesto por Criminisi (derecha) intenta preservar la forma de la línea y empieza a su alrededor.

Se puede apreciar en la figura 15 cómo el algoritmo de la media ha ido recto completamente hacia abajo mientras que los otros dos han tenido en cuenta información de la imagen. En este caso el de la cebolla ha ido recorriendo el borde de la imagen segmentada y dilatada y el algoritmo de inpainting propuesto por Criminisi ha comenzado por la zona en la que está la línea para preservarla.

Finalmente se pueden ver cada uno de estos algoritmos funcionando sobre una imagen real del problema. Con la máscara que se encuentra a continuación se han obtenido los siguientes resultados según el algoritmo empleado:

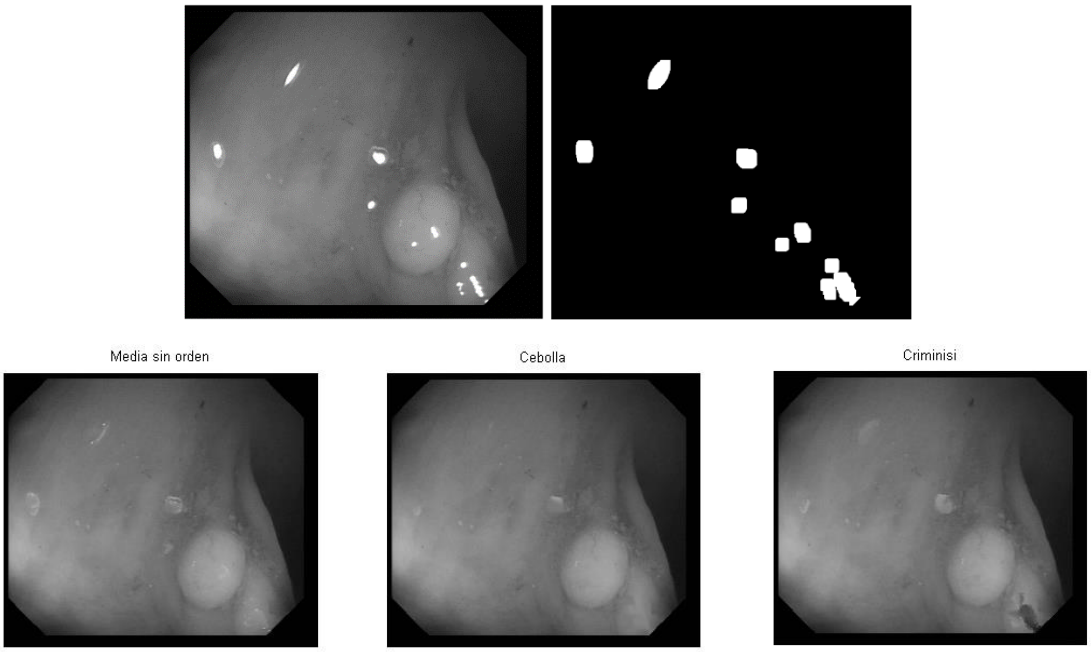


Fig. 16. Figura que muestra la imagen original (arriba a la izquierda) junto con su segmentación de los brillos (arriba a la derecha). Y cómo cada uno de los algoritmos rellena los brillos encontrados. A la izquierda el de la media, en el medio tenemos el algoritmo de la media, y a la derecha el algoritmo propuesto por Criminisi..

Las imágenes superiores muestran la imagen original a la que se le quiere aplicar el extractor de brillos, junto con la máscara que se ha calculado que indica la posición y tamaño de los brillos, y por último, muestra una comparativa de los tres tipos de extractores de brillos utilizados. Se puede apreciar claramente que el extractor de tipo cebolla ha sido el que mejores resultados ha dado. Criminisi a pesar de ser el algoritmo más complicado no nos es muy útil en este problema debido a la naturaleza de las imágenes y que lo que nos interesa no es mantener texturas del fondo, sino rellenar los brillos de modo uniforme para que no sean detectados como pólipos por su forma. Por último, el método de la media, parece que obtiene buenos resultados pero sin duda peores que el de cebolla, ya que se deja notar la posición en la que estaban los brillos. Por lo que podemos deducir que el método de la cebolla es el mejor en este problema para rellenar los brillos.

5.2.2 Corrección de píxeles negros

Otro de los grandes problemas que surgió fue que para los brillos que están en el borde no se querían tener en cuenta los píxeles negros que forman el pequeño marco negro que contienen todas las imágenes. Por lo que se hacía una comprobación para ver si el píxel era negro (intensidad 0) y en tal caso no tenerlo en cuenta para el cálculo del relleno. Sorprendentemente y a pesar de hacer esta comprobación, seguía habiendo problemas con los píxeles de los bordes. Esto era debido a que a pesar de que a la vista parece negro, debido al formato de compresión de imagen, no es negro absoluto, es decir no tiene intensidad 0, sino 0.01 o valores parecidos pero no 0 exacto. Esto hacia que se siguiesen teniendo en cuenta estos píxeles. Por lo que otro procesamiento de la imagen que hubo que hacer fue el de convertir en 0 exacto todos aquellos pixeles próximos a 0 en la imagen pertenecientes al marco.

5.3 Extracción de características

La extracción de características consiste en generar un conjunto de variables que definan las formas de la imagen. Es decir, cada ejemplo (ventana de la imagen) queremos transformarlo en un conjunto de atributos que lo represente. En nuestro caso, lo que queremos es transformar esa información en atributos que sean capaces de codificar las formas que están presentes en la ventana. Para ello se va a utilizar el extractor de características denominado HOG (Histogram of Oriented Gradients).

Este extractor de características se utiliza mucho para la detección de objetos. Consiste en ir recorriendo la imagen de la que se quieran extraer las características en ventanas de 8 pixeles de lado. Sobre esta ventana se calcula las direcciones del gradiente y se normalizan en 9 direcciones. De la manera:

Si la dirección está entre:

0-40 grados -> 1
41-80 grados -> 2
81-1200 grados -> 3
121-160 grados -> 4

Y sucesivamente. Quedando así para la ventana de 8x8 pixeles un vector de 9 posiciones donde cada posición contiene la suma de las magnitudes del gradiente de todos los pixeles cuya dirección es la posición del vector. Es decir, para la posición 1 del vector se tiene la suma de todas las magnitudes del gradiente de los pixeles de la ventana de 8x8 cuya dirección era 1, y de la misma manera con las demás direcciones. Se hace lo mismo con otras 3 ventanas de 8x8.

Por ejemplo si disponemos de la siguiente imagen:

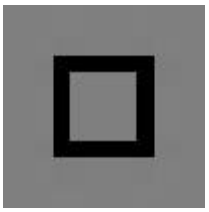


Fig. 17. Figura de ejemplo para realizar el proceso del HOG sobre ella. Es una figura de 8x8 píxeles.

Se obtienen las siguientes tablas que contienen las magnitudes y las direcciones respectivamente, es decir, el gradiente de la ventana de 8x8 píxeles. En este caso obtenemos la siguiente matriz de magnitudes.

1.41	2.00	3.16	2.83	2.83	3.16	2.00	1.41
1.41	181.02	402.87	508.00	508.00	402.87	181.02	1.41
0.00	400.34	356.38	176.80	176.80	356.38	400.34	0.00
2.83	507.01	179.61	541.64	541.64	179.61	507.01	2.83
2.83	507.01	179.61	541.64	541.64	179.61	507.01	2.83
0.00	400.34	356.38	176.80	176.80	356.38	400.34	0.00
1.41	181.02	402.87	508.00	508.00	402.87	181.02	1.41
1.41	2.00	3.16	2.83	2.83	3.16	2.00	1.41

Fig. 18. Figura que muestra la matriz de magnitudes de los gradientes asociada a la figura de ejemplo. Cada celda de la tabla corresponde a un píxel de la imagen.

Y la siguiente matriz de direcciones:

315.00	0.00	108.43	135.00	45.00	71.57	180.00	225.00
45.00	134.55	108.52	90.00	90.00	71.48	45.45	135.00
0.00	161.66	135.23	44.08	135.92	44.77	18.34	0.00
315.00	180.34	225.45	315.00	225.00	314.55	359.66	225.00
45.00	179.66	134.55	45.00	135.00	45.45	0.34	135.00
0.00	198.34	224.77	315.92	224.08	315.23	341.66	0.00
315.00	225.45	251.48	270.00	270.00	288.52	314.55	225.00
45.00	0.00	251.57	225.00	315.00	288.43	180.00	135.00

Fig. 19. Figura que muestra la matriz de direcciones de los gradientes asociada a la figura del ejemplo. Cada celda de la tabla corresponde a un píxel de la imagen.

Con estas dos matrices podemos dividir las direcciones en 9 cestas y sumar las magnitudes de las celdas que caigan en la misma cesta.

8	1	3	4	2	2	5	6
2	4	3	3	3	2	2	4
1	5	4	2	4	2	1	1
8	5	6	8	6	8	9	6
2	5	4	2	4	2	1	4
1	5	6	8	6	8	9	1
8	6	7	7	7	8	8	6
2	1	7	6	8	8	5	4

Fig. 20. En esta figura podemos ver en que cesta caen cada uno de los píxeles de la imagen en función de la dirección de su gradiente. Cada cesta corresponde a un intervalo de 40 grados.

Con estas cestas obtendríamos el siguiente histograma de gradientes orientados.

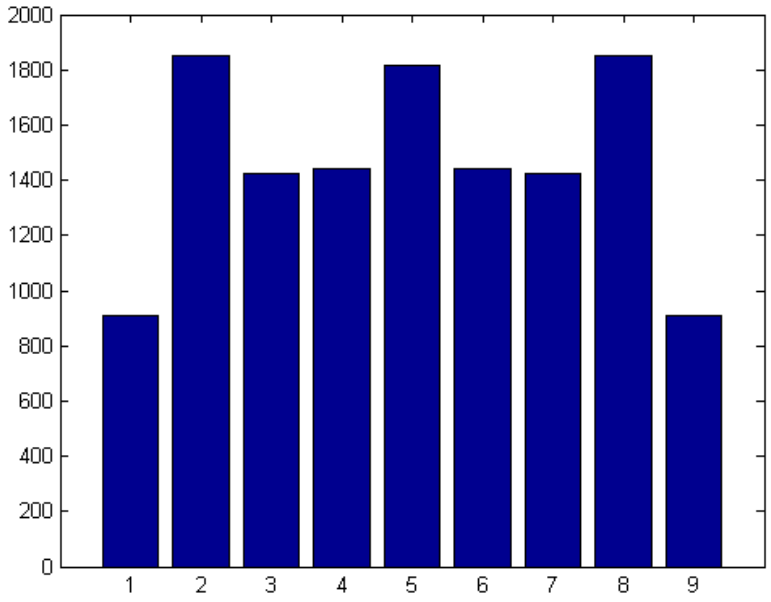


Fig. 21. En esta figura se ve el histograma de gradientes asociado a una imagen de 8x8 píxeles. Cada barra contiene la suma de las magnitudes de los píxeles asociados a la cesta en la que caen.

Y cada uno de los histogramas (vectores) que se calcula de esas ventanas se concatena. Por lo que para cada ventana de 16x16 obtenemos un vector de 36 posiciones.

Este vector se normaliza según la Norma L2:

$$\textit{L2-norm: } f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

En este método existen varias formas de normalización, pero en este proyecto se utiliza la más común. Una vez se tiene este vector se avanza al siguiente bloque de 16x16 con un solapamiento del 50% y se repite el proceso. Los vectores que se van calculando de cada bloque de 16x16 se van a su vez concatenando.

Esto es, dada una imagen dividida en bloques de 8x8 píxeles, se procede de la siguiente manera.

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24

Fig. 22. Figura que representa una pequeña imagen. Cada celda representa una ventana de 8x8 píxeles. Está numerada para poder indicar cómo se va realizando el proceso del HOG.

Si cada celda de la figura 22 corresponde a una ventana de 8x8 píxeles, la primera ventana de 16x16 sería la ventana con los índices 1, 2, 9 y 10. De esta ventana se obtiene un vector con 36 posiciones, ya que de cada una de las celdas se obtendría un vector de 9 posiciones en nuestro caso y al concatenarlos uno detrás de otro tendríamos 36 posiciones. La siguiente ventana contendría los índices 2, 3, 10 y 11. Y se obtendría otro vector con 36 posiciones que se concatenaría al anterior y así sucesivamente hasta recorrer la imagen entera. Finalmente para nuestras imágenes de 64x64 se dispondría de un vector de 1764 características. Por lo que se ha reducido una imagen de 64x64 (4096 píxeles) a un vector de 1764 posiciones de información que podemos utilizar para nuestro propósito, ya que cada posición no indica la intensidad de color, sino que el conjunto del vector contiene información de la imagen, en este caso de las formas que hay presentes.

Con respecto al método utilizado, el HOG, existen gran cantidad de variables que se pueden cambiar para que el procesado sea distinto. Pueden ser el tamaño de cada una de las celdas (en nuestro caso 8x8 píxeles), el tamaño del bloque (en nuestro caso 2x2 celdas, que al tener cada celda 8 píxeles quedaría en total 16x16 píxeles), el grado de solapamiento entre bloques (en este proyecto se utilizan 50%) y el número de direcciones que se van a tener en cuenta (nosotros utilizaremos 9 direcciones).

Estos parámetros asignados no son aleatorios y se han establecido así ya que Navneet Dalal [4] fue una de las personas que consiguieron que éste método se empezase a utilizar más, ya que lo utilizaron para encontrar personas en imágenes estáticas. En una de sus investigaciones intentaron discernir cuál era la mejor configuración para extraer las características de la imagen, y llegaron a la conclusión de que era la configuración que nosotros utilizamos. En este proyecto también se ha probado con varias configuraciones y se ha llegado a la misma conclusión.

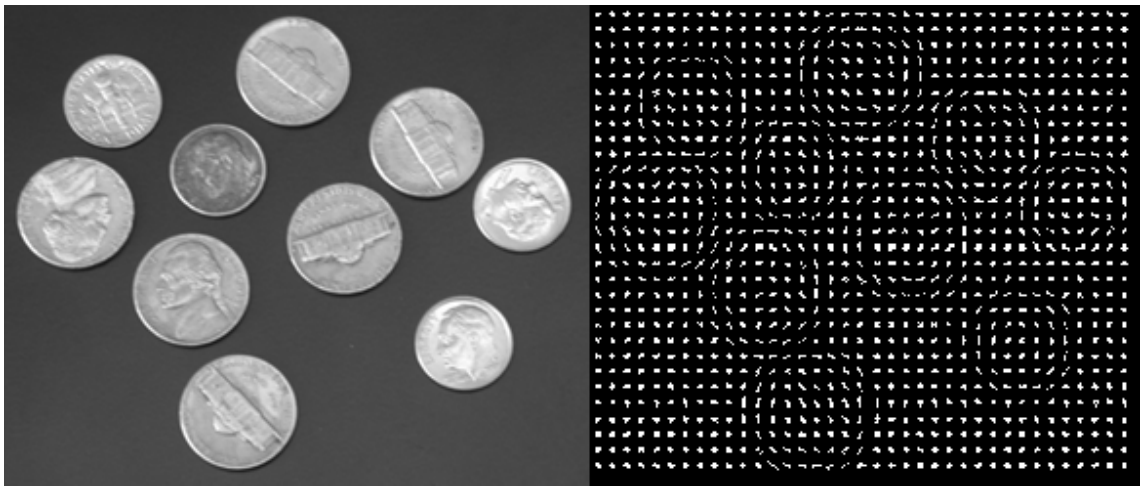


Fig. 23. Figura que muestra una imagen en escala de grises y la visualización del HOG asociado a ella que permite ver las formas que hay presentes a la izquierda. Se observan las distintas direcciones que permiten ver el lugar de las monedas.

En la figura 23, se puede ver como el método de extracción de características del HOG deja percibir claramente la posición de las monedas. En este caso, al ser redondas se puede apreciar su forma redondeada en la imagen de la derecha. Este modo de funcionamiento lo hace un caracterizador de imágenes idóneo para el uso que se le quiere dar en este proyecto, que consiste en buscar formas ovaladas.

5.4 Clasificación

Un clasificador es un algoritmo que permite asignar a un elemento entrante (que llamaremos ejemplo) una clase de pertenencia. Es decir, que introduciendo un ejemplo nos indique la clase a la que pertenece.

Una manera de realizar esto, es disponer de un conjunto amplio de ejemplos ya clasificados que nos permitan generar una regla o un conjunto de reglas para poder establecer a qué clase pertenecerán los próximos ejemplos que se le proporcionen al clasificador.

En este proyecto se van a clasificar todas las subimágenes mediante algoritmos de aprendizaje supervisado. Este tipo de aprendizaje es aquel en el que se entrena un modelo a partir de ejemplos de los que previamente conocemos el resultado que tienen que dar. Es decir, para el entrenamiento de este tipo de clasificadores se le introducen dos parámetros, los ejemplos con los que se desea entrenar junto con los resultados deseados, en nuestro caso si el ejemplo que se está introduciendo contiene pólipo o no. En este proyecto se van a usar dos algoritmos, Supporting Vector Machine (SVM) y redes neuronales.

5.4.1 SVM (Supporting Vector Machine)

Una máquina de soporte vectorial, o más comúnmente SVM (Supporting Vector Machine), es un algoritmo que trata de encontrar un hiperplano o conjunto de ellos que separe cada punto de los proporcionados en su clase. Este algoritmo funciona bastante bien para espacios de alta dimensionalidad (o incluso infinita). Trata de encontrar un modelo que represente los puntos de

muestra en el espacio y separe las clases a dos espacios lo más amplios posibles mediante el hiperplano de separación. Este hiperplano está definido como el vector entre los dos puntos de las dos clases más cercano a lo que se denomina vector de soporte.
En este caso a la SVM hay que pasarle una matriz en la que cada fila represente un ejemplo y cada columna un atributo. Junto a un vector columna en el que cada posición indica la clase a la que pertenece el ejemplo análogo.
Una vez entrenada la máquina de soporte vectorial, ésta nos proporciona un modelo, al que pasaremos ejemplos, y nos proporcionará la clase predicha para ese ejemplo.

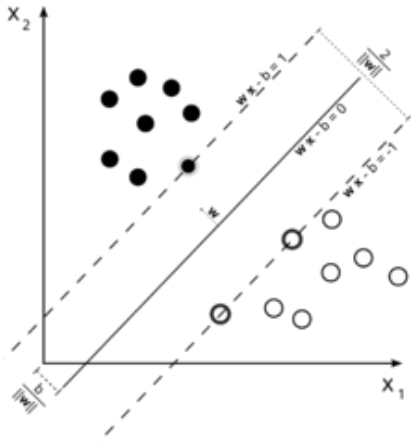


Fig. 24. En este ejemplo de SVM se puede ver el vector de soporte calculado para un espacio de dos dimensiones.

5.4.2 Redes neuronales

Una red neuronal es un algoritmo bioinspirado, que intenta imitar la red de neuronas que existe en el cerebro humano, aunque por supuesto, mucho más simple. Es un conjunto de unidades simples interconectadas entre ellas, cada unidad se denomina neurona y se comporta de forma análoga al comportamiento observado en los axones de las neuronas en el cerebro biológico. Cada unidad neuronal está conectada con todas las de la capa posterior. Cada neurona funciona de forma individual, y a partir de una entrada, normalmente en forma de vector, que le viene de la capa anterior, produce un resultado que será enviado a todas las neuronas de la siguiente capa. Este resultado puede estar limitado por alguna función límite, para que la salida sea de la forma esperada.
Este tipo de algoritmos están cada vez más al alza, ya que funcionan muy bien. Además es muy probable que casi cualquier problema de clasificación pueda resolverse mediante una red neuronal.
Si bien es cierto que este tipo de algoritmos son muy potentes, son muy complicados de entender al 100% ya que tienen multitud de factores que pueden influir en su rendimiento y precisión.
Normalmente existen tantas neuronas a la entrada como atributos tenga nuestro ejemplo. En nuestro caso tendremos 1764 neuronas de entrada, la capa intermedia (o capa oculta) variará en el número de neuronas, y tendremos una única neurona de salida. Esta neurona nos proporciona una probabilidad de pertenecer a la clase positiva, por ello, en caso de que ésta

probabilidad supere un umbral (en nuestro caso utilizaremos 0.5) diremos que la imagen de 64x64 píxeles contiene un pólipo.

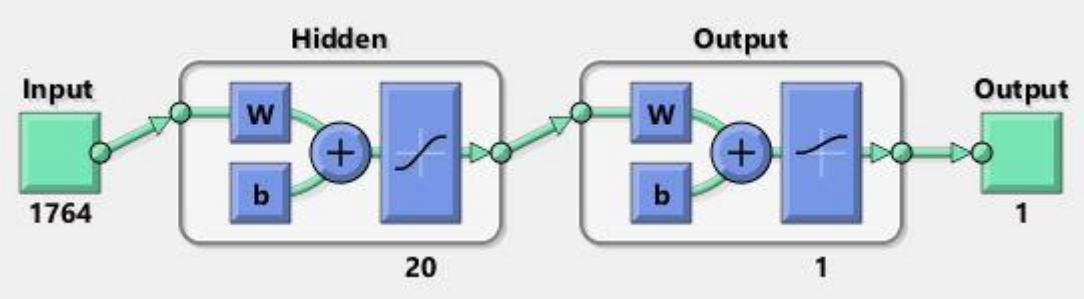


Fig. 25. Imagen de la representación de una red neuronal de nuestro proyecto. Contiene 1764 neuronas de entrada, una por cada característica. Tiene 20 neuronas en la capa oculta y una neurona de salida ya que solo nos interesa saber la probabilidad con respecto a la clase.

En la Figura superior se puede ver la configuración de una red neuronal utilizada en este proyecto, en concreto esta configuración corresponde a una red neuronal que contiene 20 neuronas en la capa oculta de esta red. Las redes neuronales de este proyecto contienen 1764 neuronas de entrada, ya que nuestro vector de características va a contener ese número de valores. Y tiene tan sólo una neurona de salida ya que si el valor de salida de la red en esta neurona es mayor que 0.5 diremos que se ha clasificado como positivo y si no como negativo. Por lo que no nos hacen falta más neuronas de salida.

Para el entrenamiento de estos clasificadores, necesitamos, como hemos mencionado anteriormente, unos ejemplos ya clasificados. Para ello utilizaremos los ejemplos que hemos obtenido en la fase de generación de la base de datos, que nos proporcionaba un directorio en el que teníamos imágenes de 64x64 píxeles que sí contenían pólipo en ellas, y otro directorio de imágenes del mismo tamaño que no contenían pólipo. Por lo que accediendo a cada directorio sabremos si su clase es positiva o negativa.

A cada una de estas imágenes se le aplica la selección de características que hemos mencionado anteriormente, el HOG. Tras aplicar esta selección de características, obtenemos un vector de 1764 características que servirá como uno de los ejemplos, junto con su clase dependiendo del directorio del que haya salido.

5.4.2.1 Selección del número de neuronas ocultas

Para seleccionar el número de neuronas a utilizar en la capa oculta que tiene la red neuronal se han probado diferentes configuraciones de redes neuronales con diferentes números de neuronas en la capa oculta. En este caso vamos a poder ver el número de verdaderos positivos, clasificaciones en las que el clasificador ha dado un resultado positivo cuando realmente es un resultado positivo. Es decir, cuando el clasificador (red neuronal en este caso) acierta.

Para este análisis se emplean los resultados de la clasificación de 100 imágenes, y un verdadero positivo indica que en esa imagen se ha detectado el pólipo en la posición correcta. Es decir, si tenemos 60 como número de verdaderos positivos quiere decir que en 60 de esas 100 imágenes se ha encontrado el pólipo en la posición real, y en las otras 40 o no se ha encontrado el pólipo o no ha sido en la posición real.

Para el entrenamiento de estas redes se han empleado 506 ejemplos, de los cuales 480 corresponden a ejemplos negativos (ventanas que no contienen pólipo) y 26 corresponden a ejemplos positivos (ventanas que sí contienen pólipo). Se puede observar que estos ejemplos están muy desbalanceados, ya que no se disponen casi de ejemplos positivos.

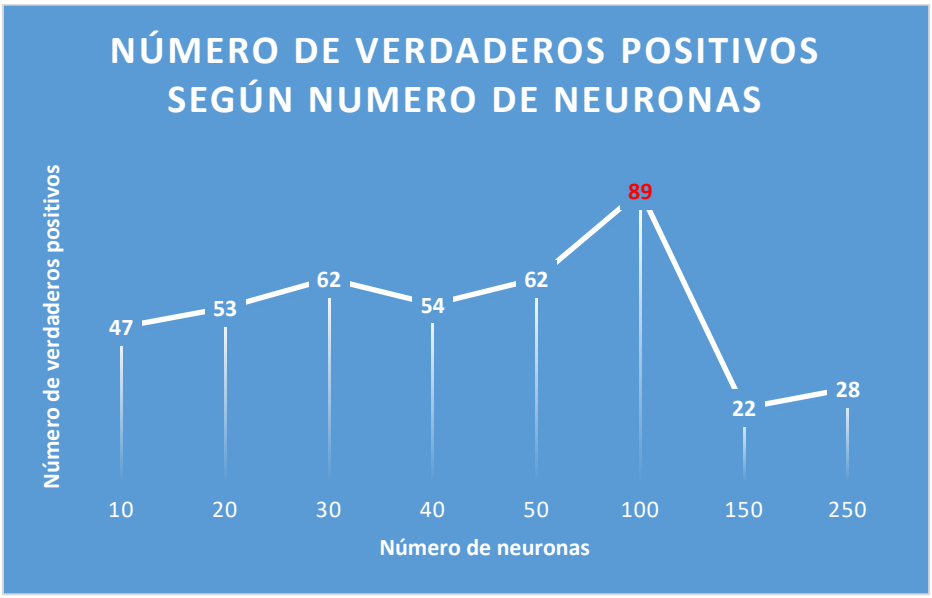


Fig. 26. Gráfica que muestra el número de imágenes en las que la red neuronal encuentra el pólipo existente en ella según el número de neuronas en la capa oculta de la red. En color rojo se puede observar el número de neuronas que mayor resultado ha dado.

Según los datos obtenidos, se puede ver que la red neuronal en cuya capa oculta utilizamos 100 neuronas es la que en más imágenes acierta.

5.5 Combinación de resultados

En este proyecto se utilizan dos clasificadores distintos, la SVM (Supporting Vector Machine) y la red neuronal. Estos clasificadores por separado pueden proporcionar respuesta al problema de clasificación. Es decir, ambos pueden por separado identificar los pólipos de la imagen, ya que ambos están entrenados mediante el mismo conjunto de entrenamiento. Y por lo tanto, ambos han sido entrenados para dar solución al problema por separado.

Cada clasificador actúa de manera independiente. Sin embargo se decidió que sería mejor combinar los resultados de tal manera que teniendo los 2 clasificadores, clasifiquen las mismas ventanas, y si ambos clasificadores proporcionan una respuesta positiva, se decidiría que esa ventana contiene pólipo.

Para esto se hacen 2 imágenes de solución distintas, cada una representando la respuesta de un clasificador. Posteriormente se recorren comprobando si, donde uno de los clasificadores ha encontrado un pólipo el otro también lo ha encontrado en el mismo punto (dejando un pequeño margen de error). Es decir, si una de las imágenes indica que alrededor de un píxel hay un pólipo se comprueba por alrededor de ese píxel si el otro clasificador también clasifica por cerca de ese punto, si es así se clasificará finalmente como pólipo.

5.6 Limpieza de puntos aislados

Tras la clasificación de todas las subventanas que se le han proporcionado a los clasificadores, se obtiene una imagen binarizada que contiene unos en los píxeles cuya ventana asociada ha sido clasificada como pólipo. En esta ventana, si la visualizamos tendremos una imagen en blanco y negro que contendrá blanco en aquellos lugares en los que se haya encontrado pólipos.

En algunas ocasiones, el clasificador da una respuesta positiva errónea, para poder establecer que en ese punto hay un pólipo realmente, se le exige al clasificador cierta confianza. Para computar esta confianza lo que se hace es contar el número de veces que el clasificador ha dado resultado positivo en una zona. Si este número de veces no llega a un umbral mínimo de veces, se considera que el clasificador no está lo suficientemente seguro de ello, por lo que se eliminan los píxeles en los que a su alrededor no se llegue a ese umbral. A continuación se muestran dos resultados antes y después del postproceso.

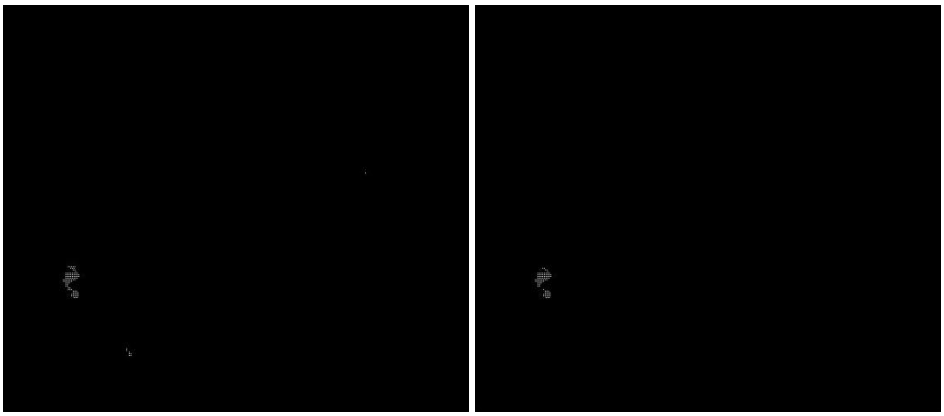


Fig. 27. En la figura se puede observar la imagen tras el proceso de clasificación que contiene píxeles blancos en los puntos en los que el clasificador ha dado resultado positivo (izquierda) y la imagen tras el proceso de limpieza de puntos aislados de la imagen anterior (derecha) que ha eliminado algunos puntos en la parte inferior izquierda.

5.7 Dibujo de los recuadros para facilitar su percepción.

Con la imagen binarizada se quieren dibujar cuadrados de 64x64 píxeles que permitan visualizar más fácilmente la posición de los pólipos encontrados en la imagen. Para ello alrededor de cada píxel clasificado como pólipo se alteran los píxeles que conformarían el borde del cuadrado. Tras este proceso se obtiene una imagen en la que se puede visualizar fácilmente la posición del pólipo localizado. A continuación se muestra la imagen de recuadros asociada a los puntos obtenidos y la misma imagen superpuesta con la imagen procesada en color.

Fig. 28. Figura que muestra la representación en forma de cuadrados para indicar la posición del pólipo presente en la imagen (izquierda) y su representación en la imagen real (derecha) que permite su mejor visualización para ayudar al médico.

6 Optimización

El problema que tenemos de la búsqueda de pólipos presentes en una imagen colorrectal, es un problema pesado, ya que hay que ir recorriendo la imagen en subventanas de 64x64 pixeles y clasificarlas mediante el clasificador entrenado con las imágenes de entrenamiento positivas y negativas de las que se dispone.

Haciendo un cálculo rápido, se ve que estamos hablando de calcular las características de la imagen mediante el HOG, que nos proporciona un vector de 1764 características que representa las formas presentes en la imagen. Pasar este vector por el clasificador entrenado para obtener un resultado para esa ventana. Y tras hacer esto con todas las ventanas se obtiene una matriz resultado que contendrá 1 donde haya sido clasificado como pólipo.

En general las imágenes de las que disponemos son de 500x574 píxeles, pero nos saltamos un pequeño hueco de unos 20 píxeles por los cuatro lados para evitar el marco de color negro, y así evitar un gran número de cálculos, por lo que recorriéndolas en ventanas de 64x64 tenemos un total de 186120 ventanas para clasificar. Esto hace que el cálculo para una sola imagen dispare el tiempo hasta más de los 8 minutos por cada imagen.

Para optimizar este proceso se han aplicado varias técnicas que van a ser explicadas a continuación.

6.1 Paralelización de bucles

El entorno utilizado para este proyecto (MatLab) utiliza tan sólo un núcleo de ejecución para hacer los cálculos. Esto se puede mejorar indicándole a MatLab en los bucles for que se quiere que haga una iteración en cada núcleo. Para ello hay que cambiar las instrucciones for por instrucciones parfor. Al hacer un bucle parfor se nos crea un trabajador de Matlab por cada núcleo que disponga el ordenador en el que se esté ejecutando y se repartirán las diferentes iteraciones entre todos los trabajadores de los que se disponga.

Para poder utilizar los bucles con parfor es necesario que cada una de las iteraciones sea totalmente independiente de la anterior, ya que al estar una iteración en cada núcleo no podrían pasarse información. Además al paralelizar un bucle mediante esta instrucción, no hay ninguna garantía de que las iteraciones de éste se hagan en un orden establecido, ya que se las dividen entre los trabajadores y cada uno las hace en un orden aparentemente aleatorio.

Con este proceso se puede reducir el tiempo de ejecución hasta un $\frac{1}{n}$ donde n es el número de trabajadores de los que se dispone. Es decir, en un ordenador con dos núcleos el tiempo no va a poder bajar más de la mitad de lo que utiliza sin la utilización de los bucles parfor.

Esta optimización se utiliza a la hora de clasificar las distintas imágenes y a la hora de generar la base de datos sin brillos. Debido a que en cada iteración no se necesita información de la anterior porque se comienza con una imagen distinta.

6.2 Número de iteraciones condicionado

Otra de las mejoras introducidas para la optimización del tiempo de ejecución es insertar una condición que permita cambiar el número de iteraciones de los bucles en los que se está recorriendo la imagen para encontrar pólipos dinámicamente. En este caso se va a emplear la información de cada momento que nos indique si existe un pólipo en la ventana que hemos procesado de 64x64 para saber si avanzar más o menos hacia adelante. En nuestro caso, si en la ventana que se acaba de procesar se ha encontrado un pólipo, nos saltaremos 1 ventana entre medio antes de avanzar, de la que no calcularemos nada y en caso de no encontrar un pólipo nos saltamos 2 ventanas. Es decir, si tras realizar la ejecución de la ventana correspondiente, nos ha dado un resultado positivo para la ventana, la siguiente ventana a procesar es la que se encuentra deslizando la ventana 2 píxeles hacia el lateral. En caso contrario, la ventana que se procesará a continuación se encuentra deslizando la ventana 3 píxeles.

Además, si al llegar al final de la fila, en ésta no se ha encontrado ninguna ventana con resultado positivo, se avanzarán 3 píxeles hacia abajo. En el caso en el que en la fila sí se haya encontrado al menos un pólipo, se avanzarán dos píxeles hacia abajo.

Todo este proceso se realiza por una razón lógica, si en una ventana no se ha encontrado un pólipo, es muy improbable que en la ventana justo posterior a ella se encuentre, por lo que intentaremos avanzar una distancia, en este caso de 3 píxeles, que sea lo suficientemente grande para que nos permita evitar un pequeño número de ventanas de las que calcular todo su proceso, pero a la vez no sea demasiado grande para que nos impida saltarnos un pólipo y no lo encontremos en la imagen.

Con esta mejora se puede notar un cambio bastante significativo en el tiempo de ejecución. Ya que se reducen notablemente el número de iteraciones que se van a realizar.

En un primero momento se realizaban 186120 iteraciones completas. Mientras que con este cambio se han reducido notablemente hasta en un 88,88% en el caso en el que en la imagen no se encuentre ningún pólipo en ninguna ventana y en un 75% en el caso en el que en absolutamente todas las ventanas se clasifiquen como pólipo. Esto quiere decir que el número de iteraciones queda acotado entre:

$$Numero\ de\ iteraciones = \ [0.1111 * n_{inicial}\ , 0.25 * n_{incial}]$$

En nuestro problema en concreto el número de iteraciones se acota entre:

$$Numero\ de\ iteraciones = [20678\ , 46530] \ll 186120\ iteraciones$$

7 Proceso completo

El primer paso del proceso es la creación de la base de datos, para ello se proporciona una carpeta con todas las imágenes, y se selecciona si el método a utilizar para la extracción de brillos va a ser el método de Criminisi o el método de la cebolla. Cuando este proceso termina se dispone de varias carpetas para utilizar, una carpeta con las imágenes en color sin procesar, una con las imágenes en blanco y negro con los brillos rellenos, otra carpeta que contiene las imágenes para entrenar positivas y otra carpeta que contiene las imágenes para entrenar negativas.

Tras este paso, se procede a entrenar los modelos, para ello se utilizan las carpetas en las que se encuentran las imágenes de entrenamiento positivas y negativas. Y al terminar este paso ya dispondremos de una SVM entrenada y lista para clasificar ejemplos, además de una matriz con los ejemplos (506 ejemplos con 1764 atributos) y un vector que indica las clases de cada ejemplo (506 ejemplos con un atributo que indica la clase) para poder entrenar si queremos cualquier otro clasificador de forma manual, en nuestro caso se han entrenado con estos datos varias redes neuronales mediante la herramienta *nntool* que MatLab permite utilizar.

El último paso es finalmente la clasificación de imágenes. Para ello se emplean las imágenes ya preprocesadas, de tal manera que no haya que preprocesar cada imagen que se va a analizar, aunque en un caso real este paso no estaría ya hecho y habría que hacerlo con cada imagen. Posteriormente se comienza a analizar desde un punto que no es el comienzo de la imagen, ya que las imágenes de las que disponemos contienen un marco negro que no queremos analizar para poder evitar bastantes cálculos.

Lo que se hace es recorrer la imagen con una ventana deslizante de 64x64 píxeles. A cada ventana se le extraen sus características, es decir se obtiene el vector de 1764 atributos y se clasifica mediante la SVM entrenada anteriormente y la red neuronal. Ambas etiquetas devueltas por los clasificadores se escriben en una matriz binaria que indicará dónde ha detectado la presencia de un pólipo cada uno de los clasificadores.

En el caso de que cualquiera de los clasificadores haya encontrado la presencia de un pólipo en la ventana a clasificar, se avanzará 2 pixeles hacia la derecha, de no ser así se avanzarán 3 píxeles. A su vez, si se ha encontrado un pólipo en toda la fila se avanzarán 2 píxeles hacia abajo, y si no 3 como se hace hacia el lateral. Este proceso se debe a que si no se ha encontrado un pólipo en una ventana es bastante posible que no esté cerca y en tal caso podemos avanzar y evitarnos en este caso el cálculo para 2 ventanas. De esta manera estamos reduciendo el cálculo desde un 75%.

Tras este segundo paso, ya tenemos las imágenes resultado, es decir, tendremos dos matrices que nos indicarán dónde ha encontrado cada clasificador la presencia de pólipos, esto es, para la entrada de esa ventana ha devuelto un 1. Antes de devolverlas y guardar los resultados, se le aplica un postproceso, de tal manera que sea un resultado más presentable. Lo primero es reducir el número de detecciones, para ello si en una zona el clasificador ha encontrado un numero bajo de píxeles como pólipo entonces diremos que se ha equivocado y los borraremos. En el caso contrario, es decir ha encontrado en una zona pequeña, muchas veces un pólipo los

mantendremos, ya que parece que ha encontrado muchas veces, por lo que puede ser que este en lo cierto. Este paso permite evitar el hecho de numerosos falsos positivos.

Estás imágenes serán el resultado del algoritmo, para mostrarlas se le hace otro pequeño postproceso para dibujar cuadrados alrededor del píxel que ha clasificado como pólipo y de esta manera superponiendo la imagen original podremos ver de una manera totalmente gráfica dónde ha encontrado pólipos el algoritmo.

Por otro lado, además se guardará la matriz resultado para poder evaluar a posteriori el rendimiento del algoritmo en conjunto.

7.1 Ejemplo ejecución

Aquí se va a mostrar un ejemplo de ejecución del proceso de manera esquemática. El primer paso es recorrer la imagen mediante subventanas de 64x64 píxeles y sacar el vector de características asociado a la subventana de 1764 características (el HOG).

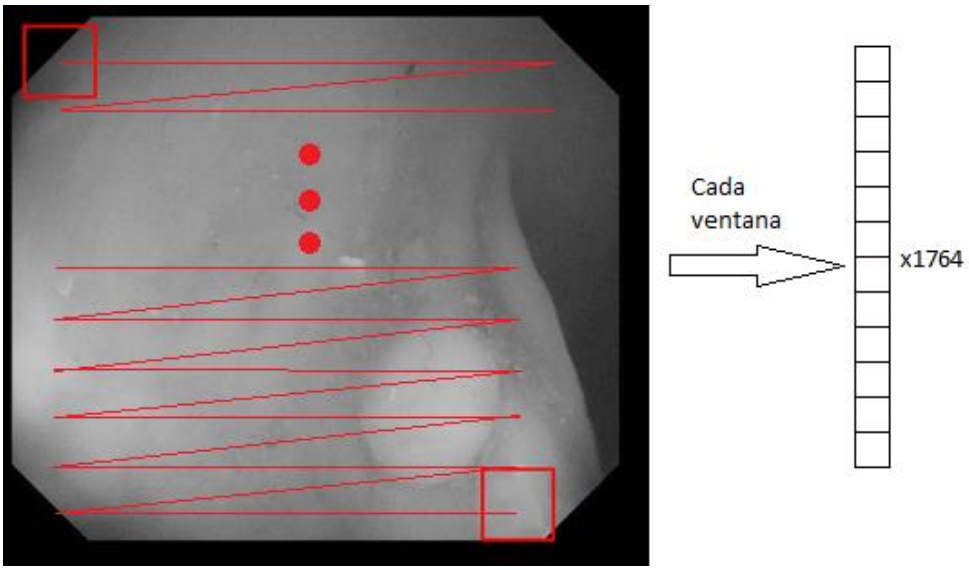


Fig. 29. En la figura se muestra el orden en el que se van clasificando las diferentes ventanas, junto con el vector de características que nos proporciona la información de las formas de cada una de las ventanas.

Cada uno de esos vectores obtenidos, se introduce tanto en la SVM entrenada como en la Red Neuronal entrenada y se obtienen los valores asociados a su clasificación por el clasificador asociado.

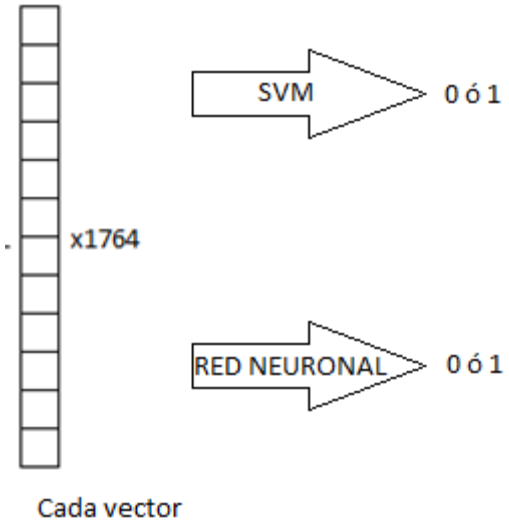


Fig. 30. Figura que muestra el proceso de introducir cada vector de características en los clasificadores pertinentes para obtener la clase predicha por cada uno de ellos.

Al pasar todas las ventanas por el clasificador se van a encontrar los posibles pólipos presentes en la imagen. Es decir, una matriz binarizada que contendrá 1 en las posiciones que los clasificadores hayan clasificado como pólipo.

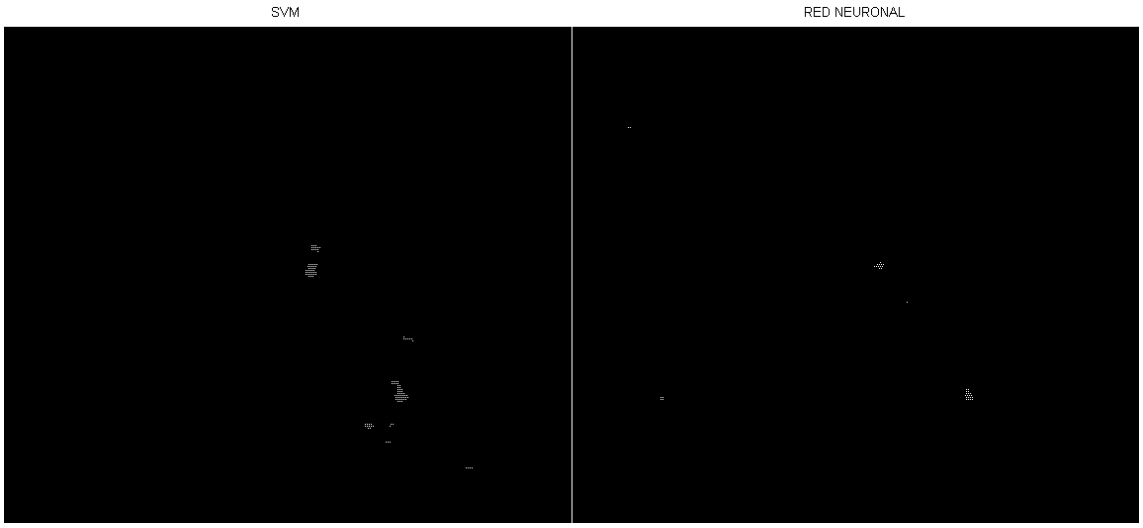


Fig. 31. Imágenes del resultado de la clasificación sin procesar que han devuelto cada uno de los clasificadores, en este caso la SVM (izquierda) y la Red neuronal (derecha).

Este es el resultado antes del post proceso para eliminar errores de clasificación. Se puede observar que en la SVM hay varias zonas con pocos resultados positivos y dos zonas con un grupo elevado de puntos clasificados como pólipos. Y en la red neuronal hay dos grupos grandes de positivos y dos grupos en los que no hay muchos puntos.

Fig. 32. Imágenes que muestran los resultados de clasificación después de aplicar un proceso de eliminación de píxeles aislados para ambos clasificadores, SVM a la izquierda y Red neuronal a la derecha.

Tras el procesado se queda de esta manera, por lo que se han evitado algunos casos donde había encontrado pólipo. Y se puede ver como se han quedado aquellas zonas donde los clasificadores habían encontrado mayor número de píxeles con resultado positivo.

Tras todo este proceso quedarían las siguientes imágenes.

Fig. 33. Figura que muestra la representación de los puntos calculados por los clasificadores y procesados para quitar los puntos aislados pero en forma de cuadrados superpuestos a la imagen original para visualizar mejor los resultados.

A partir de estas dos imágenes se combinan los resultados para obtener la imagen resultado final, que es la que ayuda al médico en el diagnóstico, dando la ubicación del pólipo de manera visual mediante el cuadrado verde que se muestra en la imagen.

Fig. 34. Resultado obtenido tras la combinación de los resultados de los dos clasificadores. En este caso la combinación ha permitido quitar una de las detecciones.

8 Resultados obtenidos

Para extraer el conjunto de imágenes sin brillo se ha utilizado el algoritmo de inpainting de la cebolla, que va rellenando las zonas marcadas como brillo por capas, desde fuera hacia el centro del brillo. Este algoritmo ha permitido crear el conjunto de entrenamiento sin que se note la posición en la que estaban los brillos antes de su rellenado. Con este conjunto de imágenes sin brillo se han extraído las muestras positivas, y junto con las muestras negativas se han entrenado los dos clasificadores utilizados en este proyecto, una SVM y una red neuronal.

En cuanto a la Support Vector Machine, se ha utilizado el algoritmo de entrenamiento y predicción que ofrece la herramienta MatLab. En nuestro caso se ha utilizado una SVM con kernel lineal, ya que es el kernel que MatLab recomienda en el caso de tener sólo una clase, como es nuestro caso (pólipo o no pólipo).

Por otro lado, con respecto a la red neuronal, se ha utilizado finalmente la red neuronal que contiene 100 neuronas en su capa oculta. Para su entrenamiento se ha utilizado también la herramienta MatLab que permite el entrenamiento de estas redes de manera muy sencilla mediante un entorno gráfico que facilita su utilización.

Para encontrar el modelo de red neuronal que se va a utilizar finalmente, se ha generado un Ground Truth, y se ha evaluado la clasificación en 100 imágenes de los tres métodos (SVM, red neuronal, y la combinación de ambos). Los resultados obtenidos se han comparado con el Ground Truth que se ha generado de forma manual, seleccionando aquellos puntos que indican el centro de los pólipos. Esto ha permitido obtener el porcentaje de verdaderos positivos y falsos positivos de cada uno de los métodos y cada una de las configuraciones. Para ello se comparan los resultados de los clasificadores con los puntos que están guardados en el Ground Truth, y si la distancia hasta el punto almacenado es lo suficientemente pequeña, se considera que el clasificador ha acertado en su predicción.

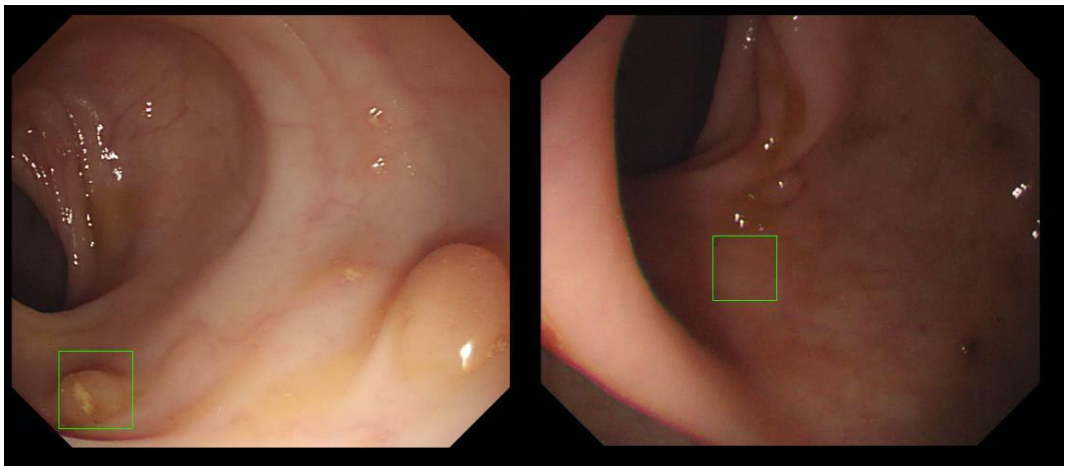


Fig.35. En esta figura se muestra el caso de un verdadero positivo (izquierda), en el que se ha encontrado el pólipo en la posición en la que está realmente. También se muestra un caso de un falso positivo y un falso negativo (derecha), ya que en la imagen de la derecha se puede observar que el lugar en el que el algoritmo ha señalado un pólipo no lo hay (falso positivo) y el lugar donde está el pólipo no lo ha señalado (falso negativo).

Tras ejecutar el algoritmo con los clasificadores que se han explicado, se han obtenido los siguientes resultados, todos ellos medidos sobre un conjunto de test de 100 imágenes.

	Verdaderos Positivos	Falsos Positivos	Falsos Negativos
SVM	45	348	55
Red 10	47	489	53
Red 20	53	709	47
Red 30	62	753	38
Red 40	54	709	46
Red 50	62	864	38
Red 100	89	2051	11
Red 150	22	134	78
Red 250	28	152	72

Fig. 36. Tabla que representa el número de clasificaciones positivas en función del clasificador y del número de neuronas. Se han comparado las clasificaciones con un Ground Truth que permite saber si son verdaderos positivos o falsos positivos.

En un principio, me decanté por la opción de 100 neuronas en la capa oculta, ya que es la configuración que mejores resultados da, sin embargo se puede apreciar que la cantidad de falsos positivos es considerablemente alta.

Debido a la naturaleza del problema, que consiste en localizar los pólipos dentro de una imagen colorrectal, decidí que es mejor maximizar el número de los verdaderos positivos antes que el porcentaje de verdaderos positivos. Esto es, si estamos evaluando sobre un conjunto de 100 imágenes, preferiremos quedarnos en este caso con aquel clasificador que aporte más número de verdaderos positivos (en nuestro caso el conjunto de clasificadores con 100 neuronas ocultas en la red neuronal), en vez de maximizar la función:

$$TPR = \frac{VP}{VP + FP}$$

Esta función calcula el número de verdaderos positivos en función de todos los positivos encontrados. Es decir, el porcentaje de los que se han clasificado como positivos de manera correcta.

En nuestro problema, se prefiere maximizar el número de verdaderos positivos antes que el ratio de verdaderos positivos, ya que al tratarse de la salud de una persona es mejor clasificar acertadamente muchos y que si el algoritmo falla e indica que se trata de un pólipo, se investigue y resulte en un fallo del algoritmo, frente a que el algoritmo falle, y no prediga un pólipo donde realmente lo hay y se quede sin investigar por un profesional médico, ya que se pondría en juego la salud del paciente.

Para dar más robustez al clasificador, se ha tratado de mezclar los resultados que dan la SVM y algunas de las redes neuronales. Esto permite eliminar unos cuantos falsos positivos, aunque también se da el caso en el que no coinciden los resultados de la SVM y la red neuronal en cuestión, haciendo que baje también el número de verdaderos positivos.

	Verdaderos Positivos	Falsos Positivos	Falsos Negativos
SVM_10	30	126	70
SVM_20	41	166	59
SVM_30	41	250	59
SVM_40	38	161	62
SVM_50	44	193	56
SVM_100	41	275	59
SVM_150	25	106	75
SVM_250	27	86	73

Fig. 37. Tabla que muestra los resultados de las diferentes combinaciones de clasificadores. En la primera columna se puede ver qué clasificadores se han mezclado, en este caso se visualiza que es la SVM junto con el número de neuronas de la red neuronal con la que se ha mezclado.

En la figura 37 se puede apreciar los resultados tras la combinación de dos clasificadores. Esto permite eliminar bastantes falsos positivos, sin embargo, también elimina una considerable cantidad de verdaderos positivos. En nuestro problema, tal y como se ha comentado, no nos interesa esto ya que al quitar estos verdaderos positivos estamos haciendo que el médico pueda pasar por alto estos pólipos, por ello es mejor tener un número mayor de este tipo a costa de incrementar también el número de falsos positivos, ya que después el médico puede descartarlo, y de la otra manera quizá no se dé cuenta.

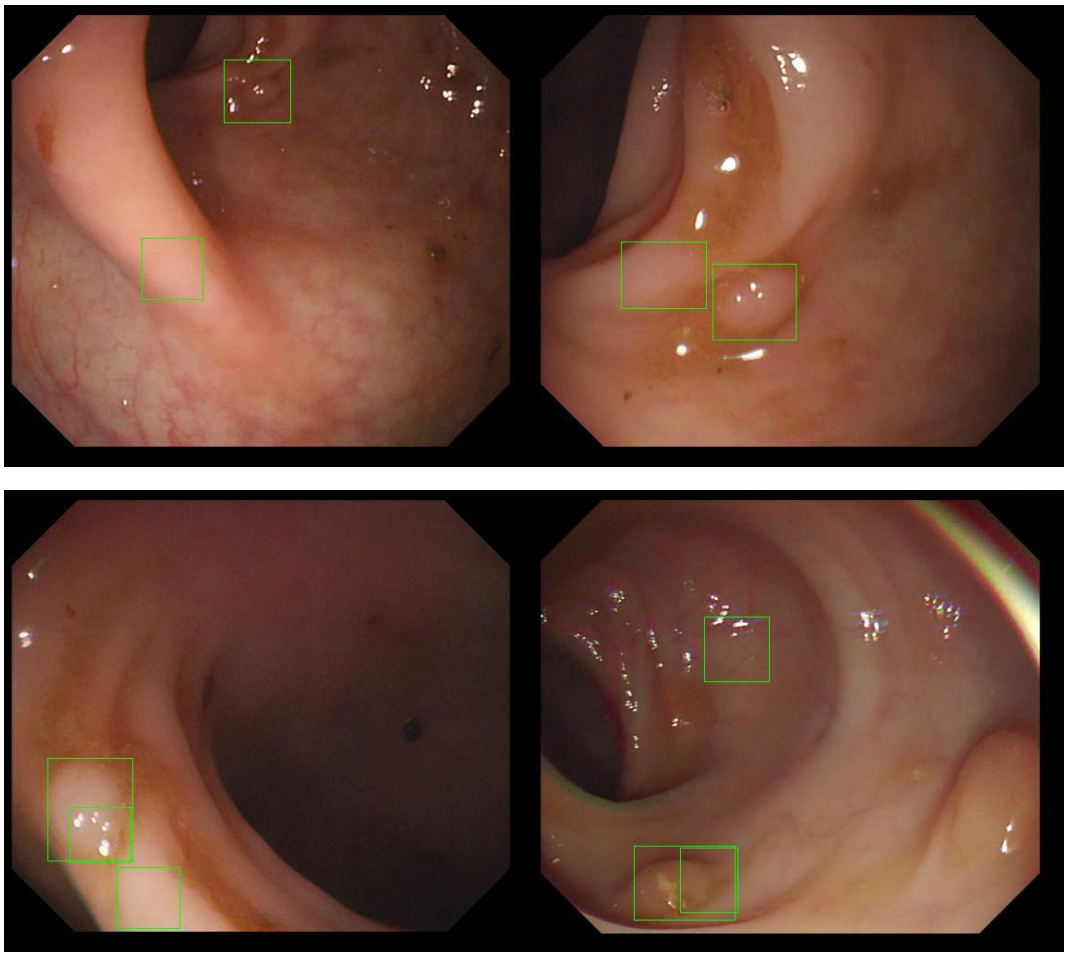
Por otro lado, aunque lo que queremos es maximizar el número de verdaderos positivos, se ha comprobado que la red neuronal con 100 neuronas en su capa oculta obtiene muy buenos resultados pero también proporciona demasiados falsos positivos que hacen que sea muy difícil discernir el pólipo en la imagen entre tantas detecciones. Esto ha hecho que finalmente escojamos la segunda mejor configuración que proporciona menos verdaderos positivos, sin

embargo proporciona también muchos menos falsos positivos. Esta configuración es la de 50 neuronas en su capa oculta.

8.1 Ejemplos de resultados

Finalmente se ha utilizado la configuración de la red neuronal con 50 neuronas en su capa oculta, ya que ha dado un alto número de verdaderos positivos sin aumentar en exceso el número de falsos positivos de la imagen.

Tras obtener los resultados se puede observar que hay multitud de imágenes que nuestro algoritmo clasifica de manera bastante precisa, es decir, encontrando el pólipo y añadiendo muy pocos falsos positivos.



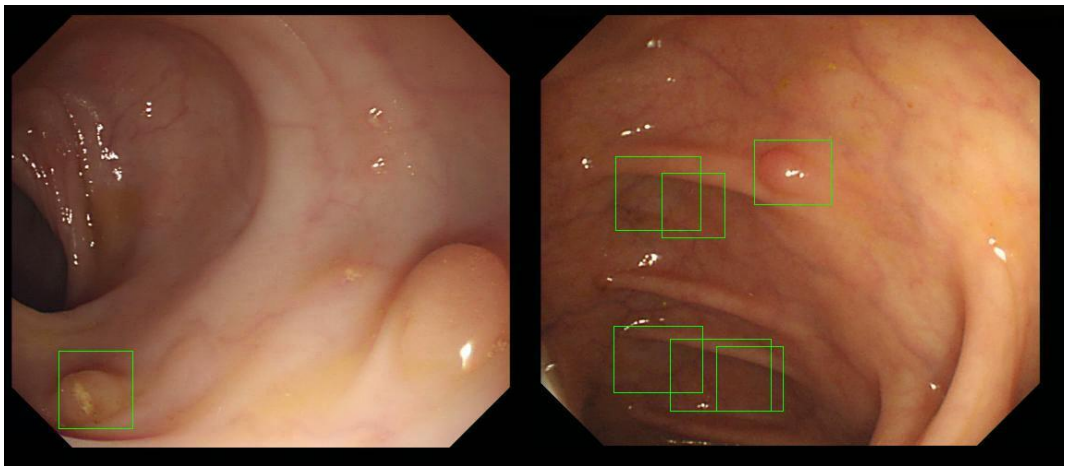
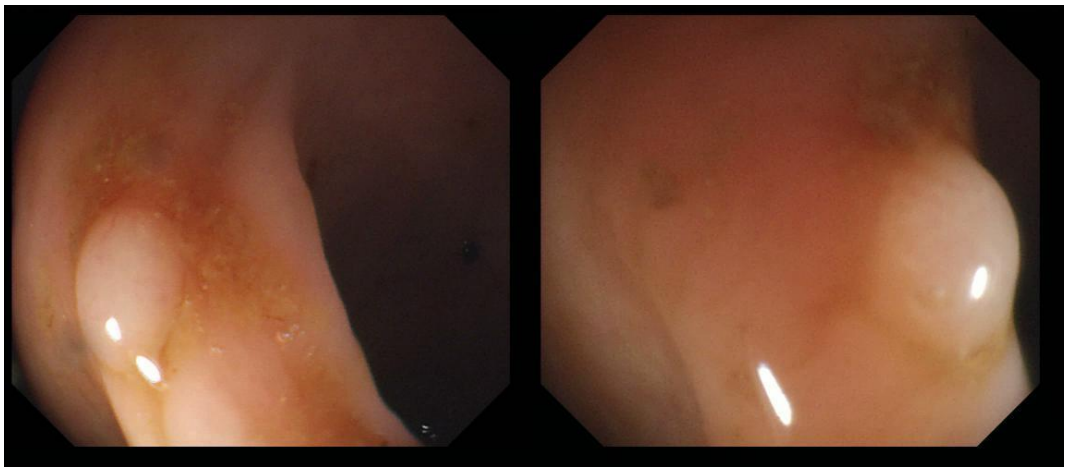


Fig. 38. Conjunto de 6 imágenes en las que el pólipo presente en la imagen se ha detectado y no ha tenido además muchos casos de falsos positivos. Imágenes como las mostradas en esta figura son el resultado ideal del algoritmo.

En la figura 38 se puede observar que ha encontrado el pólipo que hay presente en la imagen de una manera bastante precisa, aunque en algunas de ellas ha incluido algunas detecciones incorrectas (falsos positivos). En este tipo de imágenes se puede apreciar que están bastante nítidas, y no ocurre lo mismo en todas las que se han analizado. Al estar tan nítidas, las formas que hay presentes en cada subventana se acentúan, dando así lugar a mejores resultados.

Por otro lado existen otro tipo de resultados en los que no se ha obtenido un buen resultado debido a que el pólipo que aparece en la imagen es más grande que nuestra ventana de detección, por lo que con esta ventana nos es imposible encontrarlos.



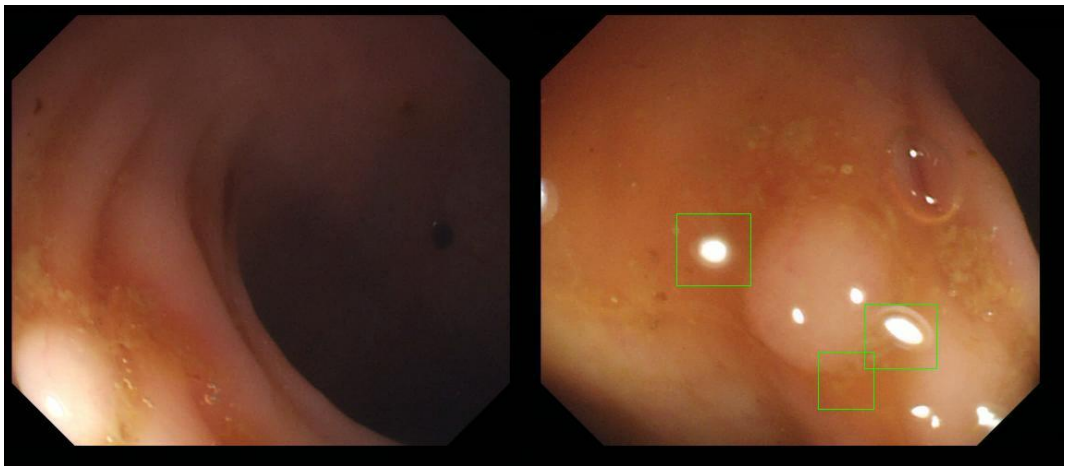


Fig. 39. Conjunto de imágenes de muestra en las cuales el pólipo es más grande que la ventana que se utiliza para buscar, por lo que no es posible encontrar el pólipo con tal ventana.

Este es sobre todo el tipo de imágenes en el que el algoritmo ha dado un falso negativo, ya que muchos de los falsos negativos se deben a este caso. Este problema es solucionable, y se ha pensado en una posible solución para mejorar en este tipo de casos. Esta solución consiste en escalar la imagen a la mitad y volver a pasar el algoritmo, ya que al reducir la imagen es posible que el pólipo sí que entre en la ventana de detección y por lo tanto, el algoritmo sea capaz de encontrarlo.

Otro de los casos posibles, es que el algoritmo detecte como pólipo una zona en la que tengamos un brillo. Esto puede deberse a que hace falta mejorar el algoritmo de rellenado de brillos un poco más. Aunque este caso no se da en gran cantidad de imágenes, hay un pequeño número de ellas que presentan errores de clasificación por este motivo.

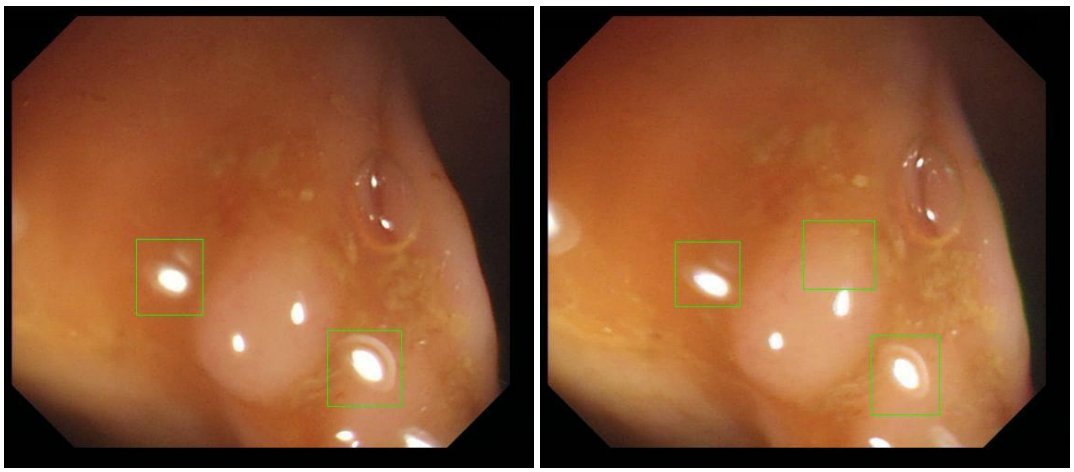


Fig. 40. Conjunto de imágenes en las que los fallos se deben a una mala gestión de los brillos, haciendo que en los brillos se produzcan unos falsos positivos.

El último caso posible de error es que el clasificador de un error de predicción, esto se debe a que el modelo no es perfecto, y aprende unas características para predecir, que no incluyen todos los posibles casos, por lo que se deja cierto margen. De esta manera se evita el sobre

aprendizaje y permite generalizar las predicciones a otro tipo de ejemplos que no son los mismos con los que se ha entrenado.

Se observa que el algoritmo ha conseguido unos resultados en los que se ha llegado a un número de verdaderos positivos de 62. Este resultado es un resultado mejorable pero sin duda es un buen resultado como primera aproximación. Viendo los resultados obtenidos se puede llegar a la conclusión de que mejorando el algoritmo de extracción de brillos y escalando las imágenes para poder encontrar los pólipos que son más grandes que la ventana que se utiliza se puede incrementar considerablemente el número de detecciones.

Estos resultados son unos resultados muy buenos para la muestra de la que se disponía, la cual está desbalanceada y no ha permitido el entrenamiento de un clasificador con mucha varianza.

Por último, se podría utilizar otro tipo de clasificadores para el entrenamiento y clasificación y observar los resultados que nos ofrecen, ya que la combinación de resultados de clasificadores, con estos clasificadores, no ha sido fructuosa y no ha conseguido aportar los resultados que se esperaba de ella.

9 Conclusiones y Líneas futuras

Tras la realización y observación de estos resultados, se puede llegar a una conclusión: es necesaria la utilización de una base de datos bastante más grande, ya que al contener mayor número de muestras y mayor diversidad creo que se hubieran alcanzado mejores resultados. Esta base de datos contenía tan sólo la secuencia de 15 colonoscopias y no todas las imágenes están muy enfocadas, además de poseer múltiples brillos que hacen más difícil su clasificación. En este caso se ha entrenado la Máquina de soporte vectorial con tan solo 26 muestras positivas y 480 negativas. Al ser tan desbalanceados estos números, no se ha conseguido entrenar un buen modelo que permita una mejor clasificación. A pesar de todo esto se han conseguido resultados bastante prometedores ya que con un modelo entrenado con muy poca varianza de datos los resultados no han sido tan malos como cabría esperar.

Los primeros resultados mostraban unas imágenes en las que casi en su totalidad de ventanas posibles se clasificaban como pólipos. Esto hacía que se produjesen una gran cantidad de falsos positivos. Los últimos resultados también dan muchos falsos positivos, sin embargo son bastantes menos que al principio, y podrían ser solucionados con una mejor base de datos de imágenes colorrectales.

Por otro lado, el problema de los brillos en las imágenes y su rellenado para poder conseguir imágenes sin brillos podía parecer a primera vista un problema no tan influyente. Sin embargo a medida que se iban obteniendo los primeros resultados, nos dimos cuenta que este problema era el factor clave a la hora de obtener mejores resultados. Y ha sido el problema en el que más tiempo se ha invertido prácticamente.

También podemos concluir con respecto a los clasificadores que se observa que la SVM no es lo suficientemente potente para este problema, arrojando peores resultados que varias redes neuronales.

Una posible línea futura que se ha quedado abierta es la utilización de redes neuronales Convolucionales para este proyecto. Este tipo de redes son un tipo de red neuronal donde las

neuronas corresponden a campos receptivos de manera muy similar a las neuronas de la corteza visual primaria. Son una variación del perceptrón multicapa, pero debido a su peculiaridad se hacen especialmente efectivas para la visión artificial, en concreto para la clasificación de imágenes. Este tipo de redes son especialmente rápidas para la clasificación de imágenes, ya que reciben de entrada la matriz entera correspondiente a la imagen que se quiere procesar y permiten cierta variación en la entrada, en nuestro caso permitirían variación en la posición del pólipo.

Otra de las líneas futuras es la realización de este mismo proyecto pero utilizando las imágenes en color, ya que en este proyecto se han transformado a escala de grises haciendo que se pierda bastante información y en color se dispone de más información. Esto abre una línea para comprobar la mejora que puede existir en tal caso.

Una posible línea de mejora es la investigación de generación de buena muestra sintética para este proyecto, debido a que en este proyecto se ha intentado pero no se han logrado resultados lo suficientemente buenos para su utilización, aplicando distintos filtros a las imágenes. En caso de no disponer de una mejor base de datos hay que conseguir más muestra para poder conseguir una mejora en el entrenamiento de los clasificadores y esta mejora es posible sólo gracias a aumentar la muestra que se les proporciona para el entrenamiento de estos. La generación de muestra sintética es un proceso complicado ya que hay que generar nueva muestra que proporcione información tan solo a partir de la muestra disponible y la información que ésta nos da.

Por último, una de las líneas futuras más interesantes es el poder conseguir los resultados totalmente en tiempo real, ya que esto abre un gran abanico de posibilidades y proporcionaría al médico que realiza la inspección unos resultados al instante y ayudaría a poder visualizarlo en el momento en el que se está realizando el examen visual a las imágenes. Este proceso se podría conseguir optimizando el código aún más de lo que se ha conseguido en este proyecto o mediante la utilización de las redes neuronales Convolucionales.

10 Bibliografía

Web de la base de datos CVC-ColonDB: mv.cvc.uab.es/projects/colon-qa/cvccolondb

Computer Vision. Linda G. Shapiro and C. Stockman. Pearson, 2001.

[1] “Face recognition using Histograms of Oriented Gradients” O. Déniz, G. Bueno, J. Salido, F. De la Torre. Pattern Recognition Letters. Vol. 32. 1 SEP 2011.

[2] “Pedestrian Detection using Infrared images and Histograms of Oriented Gradients” F.Suard, A. Rakotomamonjy, A. Bensrhair and A.Broggi. Intelligent Vehicles Symposium, 2006. Pages: 206-212.

[3] “Region Filling and Object Removal by Exemplar-Based Image Inpainting” A. Criminisi, P. Pérez and K. Toyama. IEEE Transactions on image processing. Vol.13, No 9. 9 SEP 2014.

[4] “Histograms of Oriented Gradients for Human Detection” Navneet Dalal and Bill Triggs. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.